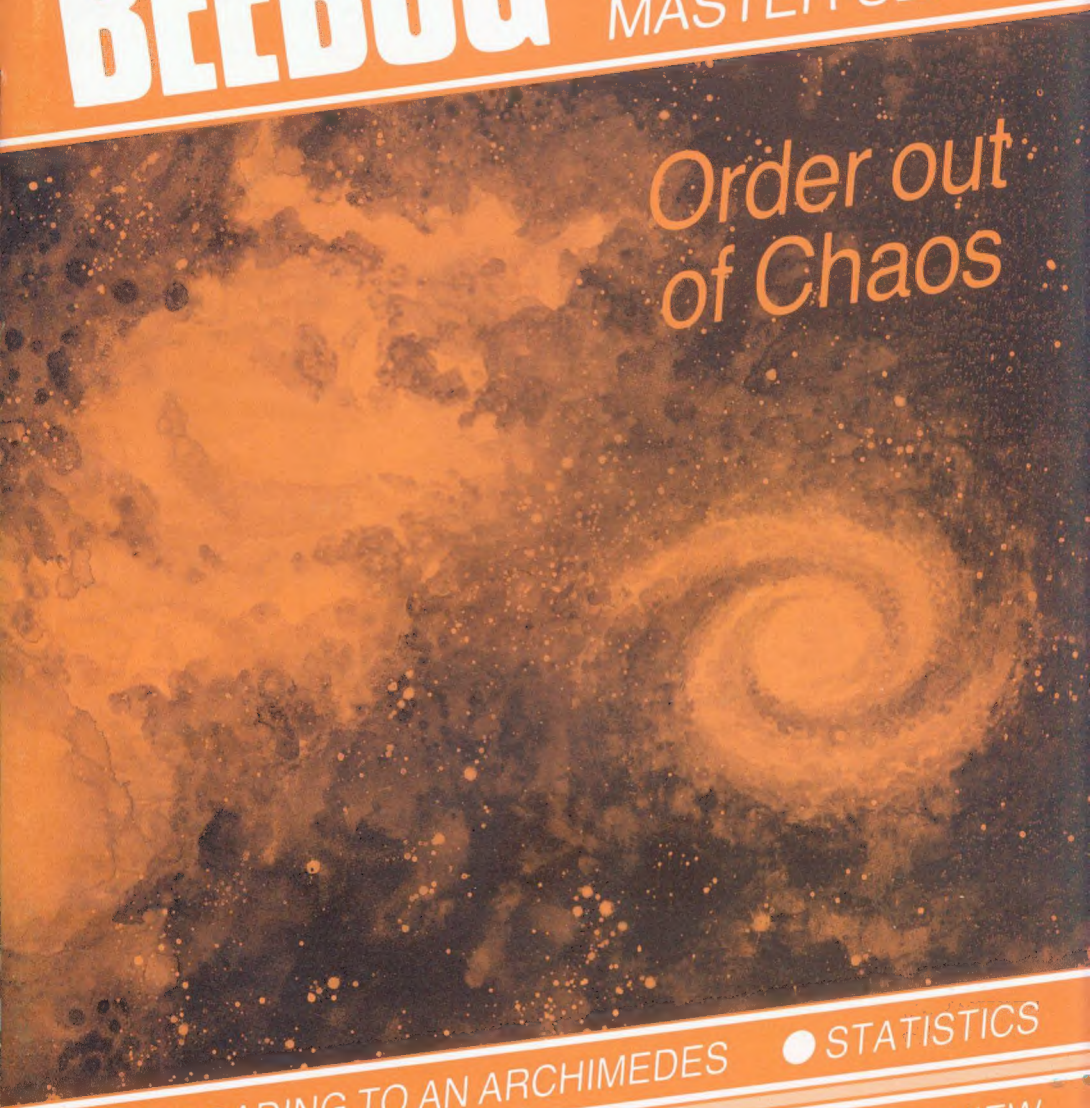
Vol.8 No.9 March 1990

BEEBUG

FOR THE BBC MICRO & MASTER SERIES



Order out of Chaos

- UPGRADING TO AN ARCHIMEDES
- STATISTICS
- KEYWORD HIGHLIGHTER
- SOFT KEYS IN VIEW

BEEBUG Vol.8 No.9 March 1990

FEATURES

Upgrading to an Archimedes 6
Keyword Highlighter 10
Order Out Of Chaos 13
Soft Function Keys Within View 18
Dichotomous Keys (Part 2) 20
Using the ROM Filing System (Part 2) 24
Statistics for Pleasure 28
Printing Characters 128 to 255 32
First Course - Improving Basic Programs (Part 2) 34
EdiKit (Part 3) 37
512 Forum 42
Workshop - Writing a Compiler (Part 4) 45
Master Display ROM (Part 2) 48
BEEBUG Education 54

REVIEWS

Games Review 23
Learning about Chaos 31
Cable Analyser Review 51

REGULAR ITEMS

Editor's Jottings 4
News 5
RISC User 53
Points Arising 56
Postbag 57
Best of BEEBUG 58
Hints and Tips 59
Personal Ads 60
Subscriptions & Back Issues 62
Magazine Disc/Cassette 63

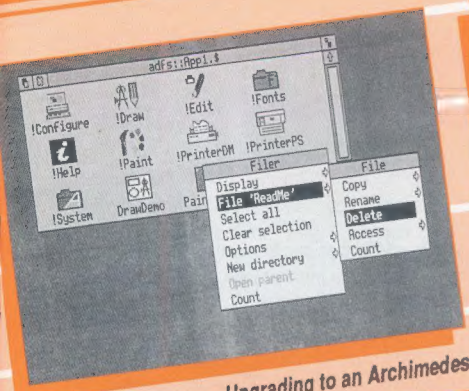
HINTS & TIPS

Hard Copy Catalogue in Wordwise
Protecting View Files on a Master
Converting a Nibble to a Byte
Quick Circle

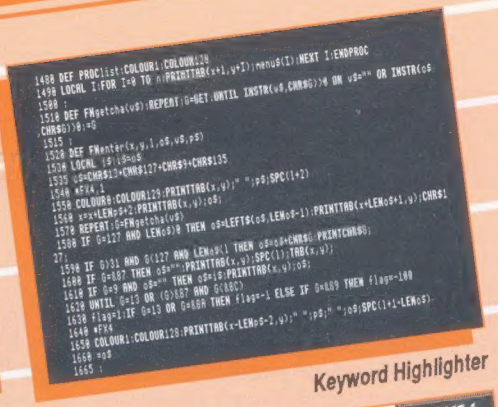
PROGRAM INFORMATION

All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

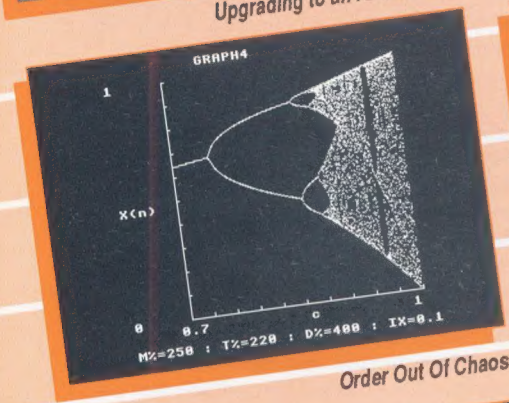
All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



Upgrading to an Archimedes

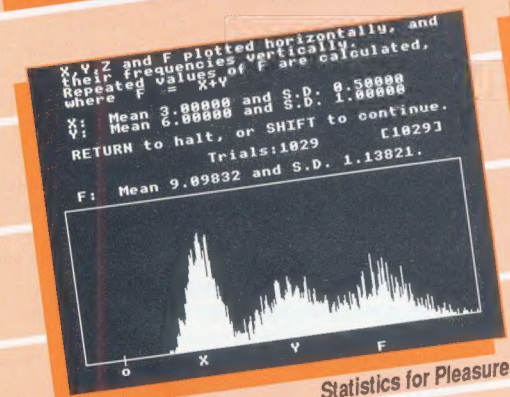


Keyword Highlighter

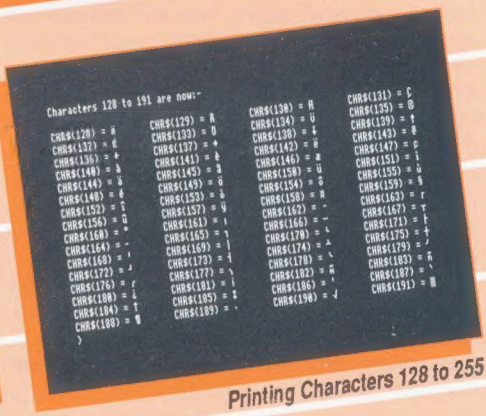


Order Out Of Chaos

Games Review



Statistics for Pleasure



Printing Characters 128 to 255

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

THE ARCHIMEDES

You will find that this issue of BEEBUG contains the first of two articles devoted to the problems, questions and decisions facing any BBC micro owner who is contemplating making the change to an Archimedes. This is in direct response to letters which we have received from readers, some of which have appeared in our *Postbag* pages. We feel that this is an important subject which is likely to be of interest to many owners of BBC micros, and thus quite a legitimate topic for BEEBUG.

It does not mean that we are in any way intending to dilute the future content of BEEBUG by catering for the Archimedes as well. Indeed, as we have said previously, the strength of BEEBUG must surely lie in the weight of support which it continues to provide for BBC micro owners, in sad contrast to many other magazines whose interest seems increasingly to focus more and more on the Archimedes. RISC User, BEEBUG's sister magazine, concentrates on that range. We hope, too, that in forthcoming issues we shall be able to provide much of interest in line with the results of the reader survey which we carried out last year.

EDUCATIONAL SUPPORT

One area where BBC micros continue to be used in large numbers is, of course, in education, and further evidence of that was forthcoming at the BETT exhibition in January where many educational computing companies had much to show that was relevant to the BBC micro.

Over the last twelve months, we have also tried to help teachers and others by publishing a variety of programs which we hope will have found a good role both at school and in the home. Programs like *Centres of Gravity, AC Circuits, Big Text Displays, Foreign Language Tester, Mathematical Transformations* and others. And many of these programs have applications outside education as well.

All of this is in addition to our bi-monthly *BEEBUG Education* which takes a broader look at the educational scene as it relates to BBC micros.

We shall be continuing this theme through 1990, witness recent programs on *Dichotomous Keys* and *Statistics*, and would welcome any further contributions for this area of the magazine.

OPEN DAYS VERSUS SHOWS

BEEBUG did not attend the BETT show referred to above. In the past we have attended many shows, particularly those which have been Acorn-specific. However, both last year and we believe this year, only a single Acorn-specific show, the BBC Acorn User Show, has been scheduled. Partly as a result, we have held two very successful open days on our own premises. We are now considering what our future policy towards shows should be, particularly with regard to the kind of presence which we should have (selling a wide range of computer products, demonstrating just our own software, etc.). We would be interested to hear the views of BEEBUG members on this issue.

MEMBERSHIP CARDS

Some members of BEEBUG have queried whether or not they have been sent the correct membership card. In fact, we use the same membership card for both BEEBUG and RISC User members. Confusion also arises because BEEBUG is both the name of this magazine, and the name of the organisation which publishes both magazines, and provides all membership services including retail, technical support, and our own software range. Membership cards show both the names BEEBUG and RISC User at the top, together with your name, membership number and expiry date (the last month of your current subscription). It is advisable to keep your card handy as it is essential to quote your membership number if asked when you deal with BEEBUG. Note that joint BEEBUG/RISC User members receive just one card.

News News News News News News

MORE SHOWS FOR ACORN FANS

Two more shows for users of Acorn computers have come to our attention. The *First Alternative European Microcomputer Show* will take place at Seeheim-Jugenheim near Darmstadt in West Germany on 28th April. The organisers promise that this will cater for Sinclair, Acorn, Cambridge Computers, Amstrad/Schneider and other "alternative" computer systems. For more information contact Stephan Michels or Dirk Schäfer at Kasinostrasse 25, D-6100 Darmstadt, West Germany.

Closer to home Acorn has announced the fifth annual computer networking conference to be held at the University of Keele from 9th to 11th July. The range of *Acorn Networks '90*, as the event is to be called, will be wider than that of previous events, and will address broader-based solutions for curriculum and administration and non-schools related applications, in addition to the usual focus on education, healthcare, special needs, business and government. For more information contact Sarah Scott at Acorn Computers Ltd, Fulbourn Road, Cherry Hinton, Cambridge, or tel. (0223) 245200.

PANDERING TO MUSICAL TASTES

After just 6 months in operation, the number of Panda music discs for Hybrid Technology's Music 5000 system has now reached a total of 13. The latest two discs, released in the new year, comprise *Moments in Time* by The Noige, whose work is well known from bulletin boards and is very much in the genre of Jean Michel Jarre. The 13th disc consists of the complete *Rendevez* suite, made famous at the Jean Michel Jarre concert in Houston and shown worldwide on television. Tracks from two other Jarre pieces, *Oxygene* and *Equinox* are also included. All Panda discs cost £5.50 inc. VAT, p&p from Panda Discs, Four Seasons, Tinkers Lane, Brewood, Stafford ST19 9DE.

IMPROVING YOUR MUSICAL OUTPUT

Hybrid Technology, the doyen of the BBC micro music world, has released a further product in the form of the *Music 5000 Synthesizer Universal* at £99. This adds high quality stereo sound to any standard music software package that might otherwise rely entirely on the built-in sound system. It provides 16 synthesized instruments, with the ability to play up to 8 of these at once (including drums). Excellent reproductions of natural instruments and 100% perfect tuning greatly improve the appeal and accuracy of any musical software, particularly appropriate in an educational environment. Fine tuning and transposition allow perfect harmony with classroom instruments, and stereo, ensemble and reverberation effects add a superb final touch for record-quality accompaniments to live performances. For more information contact Hybrid Technology, 273 The Science Park, Cambridge CB4 4WE, tel. (0223) 420360.

GAMES FOR THE BBC MICRO

Ever supportive of the BBC micro and its games market, software house Superior Software has announced two more releases, a single game *The Last Ninja 2*, and its latest compilation *Play it Again Sam 12*. Superior's policy of packaging collections of previously released games looks like catching up with itself in the near future. *Play It Again Sam 12* contains *The Last Ninja*, claimed as the number one hit game of 1989, and the forerunner of *The Last Ninja 2*, the other release.

The compilation disc also contains *By Fair Means or Foul*, a boxing simulation, *Skirmish* a jousting game, and *Blagger* a version of the classic Alligata game. Both releases are available on tape, 5.25" and 3.5" disc (£9.95, £11.95, and £14.95 respectively) covering the model B, Master 128, Master Compact and Electron (cassette only), and are reviewed in more detail elsewhere in this issue. Superior Software is at P.O.Box 6, Brigg, South Humberside DN20 9NH, tel. (0652) 57807.

ESSENTIAL SOFTWARE FOR 512 USERS

Essential Software, in which Robin Burton author of our 512 Forum has an interest, has now released further utility discs for the 512. These comprise a general purpose mouse driver at £12.95, a screen print package which operates in 40 column mode to give shaded screen dumps of four colour screens, or in 40 or 80 column mode to give a 'full speed' print of mainly text screens, costing £14.95, and a virtual screen system which prevents star commands from corrupting the DOS screen (£14.95).

All prices include VAT, p&p. All items, and further information on these and forthcoming products, are available from Essential Software, P.O.Box 5, Groby, Leicestershire LE6 0ZB.

Upgrading to an Archimedes

Mike Williams investigates the advantages and disadvantages of this major step in this two-part article.

Several BEEBUG readers have written in recently seeking advice which might help them decide whether upgrading a BBC model B or Master 128 to an Archimedes is worthwhile. Many users of such machines have often made a substantial investment in their system, and are loath to throw that away. However, an Archimedes in any form, and that includes the A3000, is such a very significant advance over the older systems that the cost may well be justified.

In this article we will be looking at what is involved, and with the help of two actual case studies, looking in some detail at what may be gained and what is lost. At the end, though, you have to make the decision on whether to upgrade based on your own priorities. We can help by clarifying possible areas of concern, and providing as much information as possible.

We shall start by summarising the main characteristics of the Archimedes, emphasizing those points where the Archimedes is distinctively different. We will then look at the compatibility between the Archimedes and older BBC micros, and the problems of attempting to transfer software. Lastly we will consider two actual case studies to highlight the potential advantages and disadvantages.

THE ARCHIMEDES RANGE

Let's be quite clear about one fact from the outset. Despite their obvious visual differences the A3000 and the Archimedes 400/1 range are all part of the same family, using the same RISC chip set and using the same operating system (RISC OS) and other software. In what follows, my remarks apply equally to *all* machines unless any differences are highlighted. The A3000 and 400/1 series represent the current models of the Archimedes, but owners of the BBC micro and Master series contemplating upgrading may well wish to consider one of the earlier A310 models (it is not worth considering an A305 machine with just 0.5Mbytes of memory unless it has been upgraded to 1Mbyte in which case it is effectively an A310).

	A310	A3000	A410/1	A420/1	A440/1
Standard Memory	1Mbyte	1Mbyte	1Mbyte	2Mbytes	4Mbytes
Maximum Acorn Memory	1Mbyte	2Mbytes	4Mbytes	4Mbytes	4Mbytes
Maximum Other Memory	4Mbytes	4Mbytes	4Mbytes	4Mbytes	4Mbytes
Floppy Disc	800K	800K	800K	800K	800K
Hard Disc	Int.	Ext.	Int.	20Mbytes	50Mbytes
Serial Interface	Yes	Opt.	Yes	Yes	Yes
Cost ex. monitor	£640.00	£746.35	£1378.85	£1953.85	£2873.85
Cost inc. monitor	*	£963.70	£1596.20	£2171.20	£3091.20

* A new colour monitor costs £217.35

Table.1 Configuration and price of Archimedes models

The basic configurations of the different models are shown in table 1 (all prices include VAT). Although 4Mbytes of memory has been the maximum memory for any Archimedes until recently, 8Mbyte upgrades are now beginning to appear (Watford Electronics). The costs shown are for a basic system, with or without colour monitor, without any additions or upgrades. The price shown for an A310 is that for a secondhand machine as supplied by BEEBUG with 3 month's warranty. New A310s are also still available at prices from £1102.85. However, there are certain differences between the A310 and the 400/1 series which might not permit the older machine to benefit from all future upgrades.

The RISC chip set is the heart of any Archimedes comprising MEMC (memory controller), VIDC (video controller), IOC (input/output controller) and the ARM processor. The A3000 and 400/1 series employ a newer version of MEMC (version 1A) which is about 12% faster than that in the 300 series. If

you are planning to buy a secondhand A310 you may wish to upgrade MEMC (which must be done by a dealer) at a cost of £75.38.

DISC DRIVES

All models are supplied with a single 3.5" internal floppy disc drive. The 420/1 and 440/1 also have a hard disc drive already fitted. The 410/1 has a built-in interface making an internal hard disc an easy upgrade. An internal hard disc may also be added to an A310, but an interface must be fitted at the same time, and a backplane (see under Interfaces for more on this). Alternatively, you can add a second 3.5" internal floppy drive to an A310 or A410.

The Archimedes series

Another route for hard disc drives is to fit a so-called SCSI interface. This permits faster and much larger capacity drives to be connected (170Mbytes for example), but these will normally be external to the Archimedes. Smaller capacity SCSI drives can also be fitted internally, except on an A3000 where the size of the casing means all hard discs must be connected externally.

MEMORY

Memory is the area which often causes the greatest concern. A 1Mbyte Archimedes will allow all current software to run, but in some cases may quite severely limit what can be achieved (e.g. the size of document to be handled by a word processor), and reduces the opportunity for multi-tasking (the facility to have several applications in operation at the same time). However, it all depends very much on what you want to do, and many owners are entirely satisfied with a 1Mbyte machine.

Potential users who are likely to make heavy demands on their machine should consider 2Mbytes, either from the outset, or as a future upgrade. Some models can only be enhanced to 4Mbytes of memory by installing non-Acorn (Watford Electronics, CJE Micros, Computerware etc.) memory upgrades. All of these are better fitted by a dealer, and with some it is essential. You should also consider carefully the effect of any such third-party upgrade on the warranty offered by Acorn. However, those aspiring to a machine such as a 440/1, but with limited finance, may find that a secondhand A310 upgraded with SCSI or ST506 hard disc and 4Mbytes of memory may prove a cheaper option, provided any problems associated with non-Acorn upgrades can be accommodated. Alternatively, the 410/1 and 420/1 can both be readily upgraded to near 440/1 specification.

INTERFACES

All Archimedes come with a printer port, an RGB analogue port for connection to a suitable monitor, a monochrome video output and a miniature stereo jack socket. All systems, except the A3000, also have a serial port fitted as standard (a 9-pin D-type socket). For the A3000 about £20 buys the two extra chips needed (the socket itself is fitted as standard).

All other forms of interface are extra and on the A310 and 400/1 series involve the use of a *backplane* (an optional extra on the A310). This fits vertically across the width of the Archimedes, and interface cards, or *podules* as they are often called, are fitted at the back of the Archimedes, automatically slotting into the backplane. Typical interfaces include the I/O podule which provides a BBC style user port, analogue port and 1MHz bus (there is also an optional MIDI interface which can be added), MIDI podule, SCSI interface, and various podules for scanners and video digitisers. There is also a ROM podule providing a ROM filing system, though this feature appears not to be popular with Archimedes owners.

The position with the A3000 is distinctively different when it comes to interfacing. In principle all is the same, but the physical shape and size of the A3000 means that the same podules as used with other models in the

Archimedes range will not fit the A3000, at least not directly. The A3000 offers two alternatives. There is a built-in mini-podule interface. At present a user port and MIDI podule (combined), and SCSI interface are available. There is also an edge connector at the rear of the A3000 which offers the possibility of connecting an interface box in which standard Archimedes podules could be fitted. At present only one such box appears to be available (from Oak Computers), but specifically for connecting a SCSI interface. Other interfaces for the A3000 are likely to appear in time.

THE ARCHIMEDES IN OPERATION

Any BBC user faced with an Archimedes for the first time is in for something of a shock. When you switch on there is no longer the customary blank screen with its simple message in the top left-hand corner. Instead you are faced with the much vaunted Desktop. Initially this consists of a uniformly grey screen with a bar (called the icon bar) running across the foot of the screen. To the left of the icon bar will be icons representing the disc drives currently installed. To the right are two other icons, the palette and the task manager. This is not the place to provide a detailed description of the Desktop and its associated functions, but a few additional words are merited.

The A3000 system

Every Archimedes has a mouse as standard, and this is the device which you rapidly become accustomed to for controlling many of the machine's functions. The Desktop uses windows and icons (in full colour) to show what is happening at all times. Click on a drive icon and a window opens to show icons representing the files and directories (the Archimedes uses only the ADFS) on that disc. Further clicking can reveal menus allowing copying, deleting, renaming of files, etc. Clicking on programs causes these to be run. All of this is controlled by the Wimp manager, a very major part of RISC OS. However daunting this may sound, or seem, at the outset, even the most experienced of Beeb users will rapidly find themselves adapting to the new ways, because they are the easiest and most efficient way of controlling the machine.

If you long for the friendly world of your BBC micro it is still there, but hidden underneath the Desktop. Quitting the Desktop can put you into Basic, the same familiar BBC Basic but with significant enhancements, and the world of star commands. Most things that you could do on a BBC micro can still be done in the same way on the Arc, but there are so many new things to learn as well. Again, the wealth of new functions and features can be off-putting initially until you get more used to them.

MEMORY USAGE

Even the Master 128 has a maximum of only 128K of memory compared to the 1Mbyte or more of the Archimedes. The contrast is even greater when you bear in mind that even on a Master, only 32K of memory exists as normal RAM, and even part of that is used by the operating system as work space; another 32K exists as shadow RAM and private RAM, and then there is sideways RAM/ROM as well. The position on the model B is even more dire, though many owners have fitted sideways and shadow RAM to enhance their systems. For comparison with the Archimedes one should consider all Beebs as 32K machines.

As a result of the limited memory available, much of the software developed for the BBC micro has been in ROM to avoid reducing even further user memory. Think how limited word processing would be if View or Interword used 16K of user RAM.

The situation on the Archimedes is quite different. Switch the machine on, and even in a 1Mbyte machine you have immediately got

some 750K of user RAM available. Think of the size of Basic program you could develop! Because of this (apparent) abundance of memory, there are no such features at all as shadow RAM, sideways RAM or whatever. All memory usage shares the same main memory (workspace, screen memory, relocatable modules - a bit like the old ROMs - and more). As a result, memory is not divided up on a fixed basis, but allocated dynamically as required. Thus your Basic program may well not always run in the same area of RAM - so no more peeks and pokes!

Any major piece of Archimedes software will normally exist as an *application*, the rough equivalent of the old ROM-based approach. Once installed, applications are allocated part of memory, and appear visibly as icons on the icon bar. Given sufficient memory, and you would be surprised how quickly even 1Mbyte can be used up, several applications can exist in memory together, each associated with one or more windows concurrently on the screen. This makes it very easy to switch from one to another. Indeed the integration of software applications is a major feature of the Archimedes.

For example, you may create data in a spreadsheet application, export part of the data into a graph package, transferring the the result to a drawing package for embellishment before incorporating the graph as an illustration on the page of your current word processed document - and it really can be as simple as it sounds, well nearly.

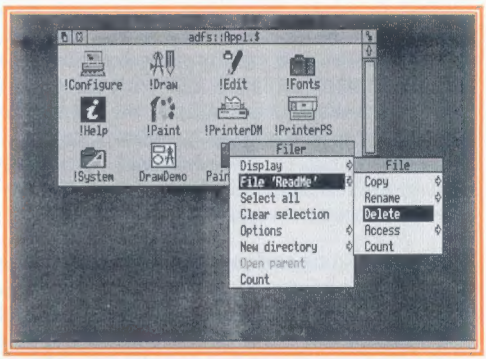
COMPATIBILITY

One of the issues that confronts anyone contemplating upgrading to an Archimedes is that of compatibility. How much of what you have, both hardware and software, can be used on an Archimedes. I hope, if I have explained things clearly enough, you will already be able to answer some of these questions for yourself now.

HARDWARE

Let's take hardware first. The Archimedes uses 3.5" discs and the ADFS. Whatever you are used to, this is what you will be using in the future. You are most likely to be using 5.25" discs with your Beeb, and most users still use the DFS. What you will probably want to do is to transfer the contents of your 5.25" discs to 3.5" discs on the Archimedes, but don't waste time and effort on things you will no longer need or use.

There are two ways of doing this. If you are able to keep your BBC system for a short while, purchase of a serial link kit from BEEBUG (cost £17.25) allows the two machines to be connected together via their serial ports (and this applies to the A3000 as well if the serial interface chips have been fitted). Files can then be transferred from one system's discs to the other's (including transfer from DFS to ADFS).



Desktop display showing Applications disc 1

Alternatively, a disc buffer can be purchased (available also for the A3000 very soon) at a cost of £33.25 allowing your existing 5.25" drive to be connected to the Archimedes. If you were previously using the DFS system you will need to purchase the BEEBUG DFS Reader (cost £9.90) for the Arc, to read discs in that format (ArcDFS from Dabs Press is another more recently available solution). Continuing to use an existing 5.25" drive can be a cheap way of providing additional disc storage for an Archimedes, and maintains compatibility with fellow Beeb users.

Any printer which you have been using will more than likely work just as well with the Archimedes (but do check with the supplier of the Archimedes, or check with BEEBUG's Technical Department), but you will need to obtain a new connecting cable as the Archimedes uses a 25 pin D-type connector.

On the other hand, any existing colour monitor is unlikely to be suitable. The Archimedes uses

Continued on page 22

Keyword Highlighter

Andrew Rowland presents a utility to help you keep up with the MacJoneses!

If you have seen advertisements for BBC Basic on the Apple Macintosh, you may have been impressed by the way listings are displayed. Keywords are shown in a bold font, variables etc. in a lighter one. Not to be outdone, I immediately began to think of a way to achieve this on the BBC itself.

The utility presented here was the outcome. In modes 0 - 6 it lists Basic programs with keywords and REMs in the normal font and everything else in a light font, while giving you the same degree of control as with the normal LIST command. This is not just for the sake of emulating the BBC's more up-market cousin; it makes a program far more readable, particularly if there are few spaces in it, or if capital letters have been used for variables.

Type in the listing and save it before running. The machine code generated by the program uses less than one page of memory, starting at &A00, although this address may be altered to suit your particular requirements by amending line 120.

Running the program installs a *LINE command which takes exactly the same parameters as LIST - that is, it can be used on its own or with the line numbers to be listed, and on a Master it even responds correctly to *LINE IF, in the same way as LIST IF (but note that the space after *LINE *must* be included). Lines can be copied using the cursor keys in the normal way. As a bonus, if you have a printer that prints italics for the codes 160 - 255, a similar effect can be obtained on paper. You may need to alter some DIP switches to get italic instead of graphic characters.

If you find that the routine interferes with some programs that use function keys, or cursor keys with Ctrl, press Break before using the program concerned, and do not run that program a second time without pressing Break again in between.

HOW IT WORKS

The first job is to create the light font, which is done by PROCfont. This creates a thin version of each character from ASCII 32 to 126, but with an ASCII code 128 higher, i.e. from 160 to 254. On a model B, this would normally require PAGE to be reset to a higher value, thus losing memory. However, assuming that PAGE is set at &1900 as is normal for a disc-based machine, we can rob the DFS of some of its workspace, and use &1300 to &15FF for the character definitions. This does mean, however, that no more than one file can be open at once and PAGE may be set no lower than &1600. The user definable characters 129 to 159 are left free for your programs.

```
1480 DEF PROC11st:COLOUR1:COLOUR129
1490 LOCAL I:FOR I=0 TO n:PRINTTAB(x+1,y+I);menu$(I):NEXT I:ENDPROC
1500 :
1510 DEF FNgetcha(u$):REPEAT:G=GET:UNTIL INSTR(u$,CHR$G))0 OR u$="" OR INSTR(u$
,CHR$G))0:=G
1515 :
1520 DEF FNenter(x,y,l,o$,v$,p$)
1530 LOCAL i$:i$=o$
1535 o$=CHR$13+CHR$127+CHR$9+CHR$135
1540 *FX4,1
1550 COLOUR0:COLOUR129:PRINTTAB(x,y);" ";p$;SPC(l+2)
1560 x=x+LEN(o$+2:PRINTTAB(x,y);o$;
1570 REPEAT:G=FNgetcha(v$)
1580 IF G=127 AND LEN(o$)0 THEN o$=LEFT$(o$,LEN(o$-1):PRINTTAB(x+LEN(o$+1,y);CHR$1
27;
1590 IF G)31 AND G(127 AND LEN(o$(l THEN o$=o$+CHR$G:PRINTCHR$G;
1600 IF G=187 THEN o$="":PRINTTAB(x,y);SPC(l);TAB(x,y);
1610 IF G=9 AND o$="" THEN o$=i$:PRINTTAB(x,y);o$;
1620 UNTIL G=13 OR (G)187 AND G(188)
1630 flag=1:IF G=13 OR G=188 THEN flag=-1 ELSE IF G=189 THEN flag=-100
1640 *FX4
1650 COLOUR1:COLOUR128:PRINTTAB(x-LEN(p$-2,y);" ";p$;" ";o$;SPC(1+l-LEN(o$)
1660 =o$
1665 :
```

A program listing showing the effect of the Keyword Highlighter

Next, we need to find a way of switching from the normal font to our new one at the right moment, without writing a large amount of code.

You are probably aware that Basic stores keywords as a single byte code, known as a token, which has a value greater than 127. As LIST displays your program on the screen, it prints the letters of variables and procedure names normally, but expands each token into the appropriate keyword. While it is doing this, it stores the value it is working on - either a normal letter or a token - at location &37. It is easy to intercept the routine which writes characters on the screen (OSWRCH) by changing the vector situated at &20E. A quick look at &37 will tell us whether a normal letter or part of a keyword expansion is being printed. PROCmc is responsible for doing this. The OSWRCH vector is diverted via *vduentry* (line

1690), and then continues by jumping to the original contents of the vector. This enables some additional code of our own to be executed before the OS carries out its normal routine. Line 1800 takes a look at &37. If it contains a number above 127, a jump to *exit* is executed, which prints the character normally. Otherwise, the character has 128 added to it (by setting the top bit in line 1870) so that it will be printed in the new light font.

There is, however, one exception to this. The contents of strings between quotation marks must be printed normally: the string may contain user defined characters or teletext control codes which should not be interpreted as keywords. In this case, the contents of &37 are not valid, and so whenever a quotation mark is detected, &37 is set to zero to ensure the whole of the string is printed in the light font (lines 1750 - 1770).

The Master treats REM lines in a similar way to characters between speech marks, but the model B treats them as normal lines (which is why teletext colour codes appear as keywords on listing). We want to ensure REMs stand out, so we set a flag, *remflag*, whenever &37 is found to contain the token for REM (line 1830). This stays set until a carriage return marking the end of the line is printed (lines 1710 - 1740), and until then everything is printed normally.

Our next task is to implement a star command to perform the listing - we hardly want the routine to be active all the time as it would play havoc with normal output. The most convenient method to use is *LINE, which is provided by the operating system specifically to run a resident machine code routine, and so the OS does all the hard work. Lines 1380 to 1670 implement the *LINE command, which first diverts the OSWRCH vector to *vduentry*, then uses the equivalent of *FX138,0,n to poke "LIST" into the keyboard buffer, followed by any parameter you typed after the *LINE. In order to keep the appearance neat, it turns off screen output while it does this: omit line 1450 to see it happening.

Next, of course, the routine must turn itself off at the right time by resetting the OSWRCH vector to its old contents. Line 1630 ensures that a zero is poked into the keyboard buffer, which is detected at line 1700 when the listing is finished. But it isn't that simple. What if you press Escape before the listing ends? The routine would be left active, and the same would happen if you typed a parameter containing an error. To deal with this eventuality, the break vector BRKV, which handles all errors, is diverted to *break* (line 2010) which restores the OSRWCH vector before passing control to Basic's error handler.

One last problem must be solved. If you were simply to use cursor editing to copy lines displayed by the routine, the program would be corrupted: all your variables would be replaced by codes above 127 and interpreted as tokens. To make the program watertight we must intercept another vector, this time the OSRDCH vector, RDCHV, which is responsible for handling cursor copying. A short piece of code at *entry* (line 1940) checks to see if the code returned by OSRDCH is 160 or above, and if so, subtracts 128 from it by resetting the top bit to zero.

Finally, line 110 provides you with a Macintosh-like paper-white screen. If you prefer the BBC's good old white on black you can omit this line.

```
   10 REM Program .>KEYhlt
   20 REM Version 1.00
   30 REM Author  Andrew Rowland
   40 REM BEEBUG  December 1989
   50 REM Program subject to copyright
   60 :
  100 VDU 14
  110 VDU 19,0,7;0;0;19,1,0;0;
  120 MC%=&A00
  130 PROCfont
  140 PROCmc
  150 PRINT'"Syntax: *LINE n,n (as LIST)
"'"Ready"
  160 END
  170 :
 1000 DEF PROCfont
 1010 IF NOT FNmaster PROCfontflags
 1020 FOR pass=0 TO 1
 1030 P%=MC%
 1040 [OPT pass*2
 1050 LDA #32:STA &80
 1060 .loop
 1070 JSR thin
 1080 INC &80:LDA &80
 1090 CMP #127:BNE loop
 1100 RTS
 1110 .thin
 1120 LDX #&80:LDY #0:LDA #10
 1130 JSR &FFF1:LDX #8
```

```
1140 .tloop
1150 LDA &80,X:LSR &80,X
1160 AND &80,X:STA &80,X
1170 DEX:BNE tloop
1180 LDA #23:JSR &FFEE
1190 LDA &80:ORA #&80
1200 JSR &FFEE:LDX #0
1210 .loop2
1220 INX:LDA &80,X:JSR &FFEE
1230 CPX #8:BNE loop2
1240 RTS
1250 ]NEXT
1260 CALL MC%
1270 ENDPROC
1280 :
1290 DEF PROCmc
1300 oswrch=&FFEE:osbyte=&FFF4
1310 quote=34:rem=&F4
1320 wrchv=&20E:brkv=&202
1330 rdchv=&210:userv=&200
1340 FOR pass = 0 TO 1
1350 P%=MC%
1360 [OPT pass*2
1370 .userentry
1380 CMP #1 \ is it *LINE?
1390 BEQ lineok
1400 JMP !userv
1410 .lineok
1420 STX &72:STY &73:\ set VDU vector
1430 LDA #vduentry MOD &100
1440 STA wrchv
1450 LDA #vduentry DIV &100
1460 STA wrchv+1:\ turn off screen
1470 LDA #21:JSR oswrch
1480 \ flush keyboard buffer
1490 LDA #21:LDX #0:JSR osbyte
1500 \ poke "L." in k/b buffer
1510 LDA #ASC"L":JSR poke
1520 LDA #ASC".":JSR poke
1530 \ poke string following
1540 \ *LINE into k/b
1550 LDY #0
1560 .strlp
1570 LDA (&72),Y
1580 CMP #13:BNE strov
1590 LDA #6:JSR poke:LDA #13
1600 .strov
1610 JSR poke
1620 INY:CMP #&0D:BNE strlp
1630 \ poke zero to
1640 \ reset VDU vec
1650 LDA #0
1660 .poke
1670 PHA:STY temp:TAY
1680 LDA #138:LDX #0:JSR osbyte
1690 LDY temp:PLA
1700 RTS
1710 .vduentry
1720 STX temp:TAX
1730 CMP #0:BEQ end
1740 CMP #32:BCS nctrl
1750 LDA #0:STA remflag:JMP exit
1760 .nctrl
1770 CMP #quote
1780 BNE over
1790 LDA #0:STA &37
1800 .over
1810 LDA remflag:BNE exit
1820 LDA &37
1830 CMP #rem:BNE notrem
1840 INC remflag
1850 .notrem
1860 CMP #128:BCS exit
1870 TXA:ORA #&80:TAX
1880 .exit
1890 TXA:LDX temp
1900 JMP (store)
1910 .end
1920 JMP unhook
1930 .entry
1940 JSR !rdchv
1950 PHP:CMP #128+32:BCC ok
1960 AND #&7F
1970 .ok
1980 PLP:RTS
1990 .break
2000 JSR unhook
2010 JMP !brkv
2020 .unhook
2030 PHA:LDA store
2040 STA wrchv
2050 LDA store+1:STA wrchv+1
2060 PLA:RTS
2070 .temp
2080 EQUB 0
2090 .remflag
2100 EQUB 0
2110 .store
2120 EQUW !wrchv
2130 ]NEXT
2140 ?userv=userentry MOD &100
2150 userv?1=userentry DIV &100
2160 ?rdchv=entry MOD &100
2170 rdchv?1=entry DIV &100
2180 ?brkv=break MOD &100
2190 brkv?1=break DIV &100
2200 ENDPROC
2210 :
2220 DEFFNmaster
2230 =INKEY-256=253 OR INKEY-256=245
2240 :
2250 DEF PROCfontflags
2260 A%=131:*FX 20,3
2270 B%=(USR(&FFF4)AND&FFFF00)/256
2280 IF PAGE<B%:PAGE=B%:CHAIN"KeyHlt"
2290 ENDPROC
```

Order Out Of Chaos

Jim Vernon explains how to investigate the world of chaos using a BBC micro

'Ordered' chaos has recently become a subject of fascination for mathematicians, and of practical value in many highly diverse fields, from turbulence in flow along pipes to chemical reactions and weather forecasting. It has links with fractals, the Mandelbrot set and Julia sets. The three programs in this article give an introduction to the subject, and can be readily expanded for more complex work.

ITERATION

The basis of this whole field of mathematics is iterative calculation, of which an early example was Newton's method for successive approximation to the roots of an equation. In its broadest terms the one-dimensional formulation is:

```
X(n+1) = f(X(n))
```

where n is 1,2,3,4 up to any number, usually large, and f(X(n)) is any function of X(n). Each approximation is calculated from the previous one, with the calculated values converging to a result. Note that X(n+1) is sometimes written X subscript n+1, but this can lead to confusion with Xn+1. X(2) is calculated from X(2)=f(X(1)), X(3)=f(X(2)) and so on, the previous value of X(n) being inserted in the function to get the new value.

The computer is ideally suited to iterative calculations. In Basic the formula becomes:

```
X=f(X),
```

for any function *f* since, when this line is acted upon, X acquires the value of the function using the previous value of X for the calculation, so that if X(n) is the old value of X the new value is X(n+1). Repeated 'n' times, by means of a FOR-NEXT or REPEAT-UNTIL loop, without any need to change the statement of the function, the iterative value of X(n) can be calculated up to any value of n.

For the purposes of this article the function used will mainly be:

```
f(X) = 4c.X.(1-X)
```

where 'c' is a constant in the range 0 to 1. To avoid confusion '.' is used to stand for multiply. In terms of the calculation of X(n+1) from X(n), the formula can be written:

```
X(n+1) = 4c.X(n).(1-X(n))
```

The starting value of X, (X(1)), must be between 0 and 1, and c also below 1. This function, known as the *logistic difference equation*, arises in population studies, where X(1) can represent the population at, say, year one, X(2) the population at year two and so on. c is some constant, determined by the environment, and not affected by population size, which represents the rate of growth. Iterated over a number of years, the formula shows how the population can rise and fall with time, other things remaining equal.

Listing 1 - the program CHAOS1 - is a straightforward simple iteration of this formula. It progressively calculates the value of X(n) as n increases, using any specified value of c (C), and with any chosen initial value of X (IX). Provision is made for the screen display to start after any particular value of n (T%) has been reached, thus making it easier to examine the value of X(n) for high values of n. The formula to be used is placed in an FN function (line 1020). The iteration is based in this case on a REPEAT-UNTIL loop (lines 210-280), and any key press will give the value of the next iteration. The process can be stopped at any value of n by pressing Escape.

Listing 1

```
  10 REM Program CHAOS1
  20 REM Version B1.2
  30 REM Author  Jim Vernon
  40 REM BEEBUG  March 1990
  50 REM Program subject to copyright
  60 :
 100 ON ERROR:PROCerror:END
 110 MODE3
 120 PRINT''"   X(n+1)=4*C*X(n)*(1-X(n)
) "
 130 PRINT''"Equation is iterated up to
 any desired value of n."
 140 PRINT'"Any initial number of itera
tions can be skipped before printing sta
rts."
 150 PRINT'"From then on press any key
for next iteration."
 160 INPUT'' "Value of constant c (<1)
",C
 170 INPUT"Initial value of X (<1)",IX
```

```
  180 INPUT "No. of initial iterations n
ot to be printed ",T%
  190 PRINT'"PRESS ANY KEY FOR A NEW CAL
CULATION OF X(n)"
  200 N%=0 :X=IX
  210 REPEAT
  220 REM 1st. T% iterations not printed
  230 IF N%>T% THEN Y=GET:PRINT"n = ";N%
;" X(n) = "X;" c = ";C
  260 N%=N%+1
  270 X=FNformula(X)
  280 UNTIL FALSE
  290 END
 1000 :
 1010 REM Reiteration equation
 1020 DEF FNformula(Z)=4*C*Z*(1-Z)
 1030 :
 1040 DEF PROCerror
 1050 VDU7:REPORT:PRINT" at line ";ERL
 1060 ENDPROC
```

Using this program it will be found that if c is less than 0.25, iteration rapidly reduces X(n) to a very small value. If c is between 0.25 and 0.75, as n increases X(n) approaches a limiting value, given by the formula 1-1/4c, whatever the starting value of X. As c is increased beyond 0.75 X(n) does not settle down to a single value. At values just in excess of 0.75 it has 2 values, alternating between them, then, at progressively higher values, it cycles through 4, 8, 16 and so on. c has to be increased by only small amounts to get these results, and preferably a large starting number for n specified. When c reaches 0.8925, X(n) becomes chaotic, with a wide range of apparently random values.

Progress towards the limiting value for X(n), when it occurs, is illustrated graphically by listing 2 - program CHAOS2. In this, up to 3 starting values of X between 0 and 1 can be specified with any one value of c in the range 0 to 1. The number of iterations can also be varied, the larger the number the more certain it is that the limiting value will be reached, but the less clear are any early fluctuations. After input of the start values (lines 180-240), axes are drawn, with the value of n along the bottom and X(n) vertically (lines 280-310). To the left of the diagram, the limit value of X(n), calculated by the formula 1-1/4c, is drawn (lines 330-360). The iterations are calculated and plotted by the procedure in line 430, the formula again being defined in a function.

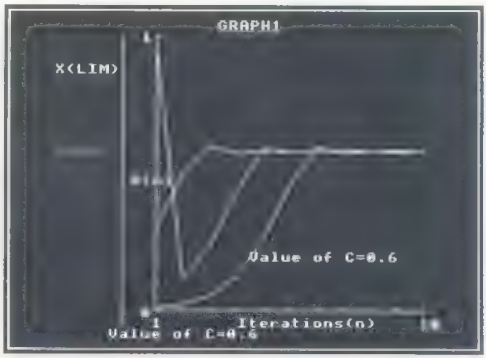
Listing 2

```
   10 REM Program CHAOS2
   20 REM Version B1.2
   30 REM Author  Jim Vernon
   40 REM BEEBUG  March 1990
   50 REM Program subject to copyright
   60 :
  100 ON ERROR:MODE7:PROCerror:END
  110 MODE1
  120 PRINT'"  X(n+1)=4*C*X(n)*(1-X(n))"
  130 PRINT'"This program graphs the val
ue of X(n)"'"against the number of itera
tions (n)."'"Opportunity is provided to
use up to"'"3 different start values of
X."
  140 PRINT'"It also shows the limit val
ue of X(n)"'"for large n."
  150 PRINT'"c can be varied to show the
 effects on"'"the limit value of X. High
er values"'"of c above 0.75 show the inc
rease in"'"number of X's limits."
  160 DIM IX(3):DIMK(3)
  170 REM Input constants, start value(s
) etc.
  180 INPUT'"Value of constant c",C
  190 INPUT"Number of iterations to be s
hown",D
  200 INPUT"Number of graphs(1 to 3)",G%
  210 FOR N% =1 TO G%
  220 PRINT"Graph ";N%;: INPUT "-Start v
alue of X",IX(N%)
  230 NEXT
  240 INPUT"Name of File",A$
  250 CLS
  260 REM Graph title, axes (with 10 div
isions) and ranges
  270 PRINT TAB(15,1);A$
  280 PROCaxis(300,150,300,950)
  290 FOR Q%=1 TO 10: PROCaxis(300,150+Q
%*80,310,150+Q%*80) :NEXT
  300 PROCaxis(300,150,1100,150)
  310 FOR Q%=1 TO 10: PROCaxis(300+Q%*80
,150,300+Q%*80,160) :NEXT
  320 REM Graph of limit value
  330 XLIM=1-1/(4*C)
  340 PRINTTAB(0,5)"X(LIM)"
  350 MOVE0, XLIM*800+150: DRAW150,XLIM*
800+150
  360 PROCaxis(200,100,200,950)
  370 REM Insert ranges of values
  380 PRINTTAB(18,22)"Value of C=";C
  390 PRINTTAB(8,2);"1":PRINTTAB(7,15)"X
(n)":PRINTTAB(8,27);"0"
  400 PRINTTAB(9,28);"1":PRINTTAB(17,28)
"Iterations(n)":PRINTTAB(34,28);D
```

```
  410 REM Calculate X(n) for each n and
graph values
  420 FOR N%=1 TO G% :K(N%)=N%
  430 PROCgraph(IX(N%),K(N%))
  440 REM Delay to allow first graph to
be viewed
  450 FOR Q%=1 TO 5000:NEXT
  460 REM Repeat for graphs 2 and 3
  470 NEXT
  480 OSCLI("SAVE "+A$+" 3000 8000")
  490 END
 1000 :
 1010 REM FN defines formula for reitera
tion
 1020 DEF FNformula(Z)=4*C*Z*(1-Z)
 1030 :
 1040 DEF PROCgraph(XP,K)
 1050 GCOL 0,K
 1060 REM XP is start value of X, K dete
rmines colour
 1070 MOVE 300,XP*800+150
 1080 X=XP
 1090 FOR M=1 TO D
 1100 X=FNformula(X)
 1110 Y=(X)*800+150
 1120 R=M *800/D+300
 1130 DRAW R,Y
 1140 NEXT
 1150 ENDPROC
 1160 :
 1170 DEF PROCaxis(A%,B%,G%,H%)
 1180 MOVE A%,B%
 1190 DRAW G%,H%
 1200 ENDPROC
 1210 :
 1220 DEF PROCerror
 1230 VDU7:REPORT:PRINT" at line ";ERL
 1240 ENDPROC
```

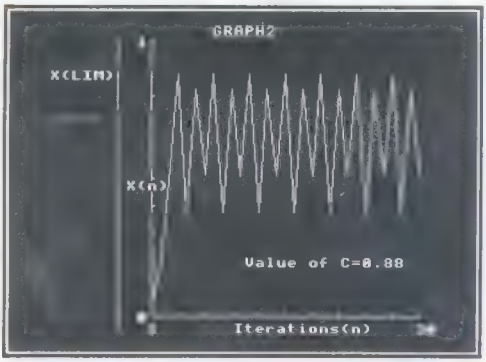
Running the program with 3 values of X, it can be seen that whatever the start value of X the graph eventually reaches the same limit, for any fixed value of c between 0.25 and 0.75. Figure 1 is a typical example. Beyond c=0.75, using only one start value of X, the graphs show how X(n) moves between a number of values - 4 in the case of figure 2. However, listing 3 - program CHAOS3 - gives a clearer picture of what is happening as c increases, by graphing X(n) directly against values of c.

In the program CHAOS3, after input of initial values (lines 170 - 230) and the drawing of the



Population converges to a steady state

axes (lines 260 - 330), the input values are also printed to provide a permanent record. The chosen range of values of c (C0,C1) is then divided into the specified number of points on the c axis (D%) and for each value the iterative values of X(n) are calculated, up to the chosen limit for n (M%), again using an FN function (lines 350 - 390).



Population oscillates between four levels

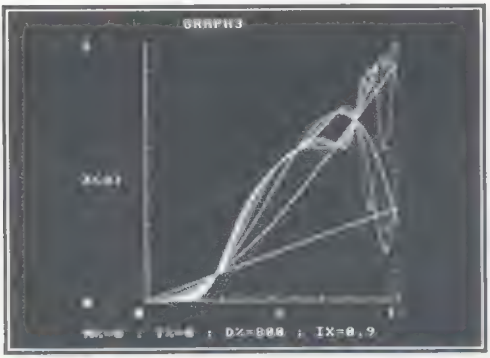
Provision is made for the first T% values to be ignored for graphing purposes, and the remainder are plotted on the vertical axis, provided they fall within the range of X(n) selected (X0,X1) (lines 410 - 480). All the values from iterating X(n) between T% and M% are thus plotted against that particular value of c. There could be many such values for any one value of c, though as we would expect for values of c between 0.25 and 0.75 there is often only a single value, so that a single point is

printed (M%-T%) times on the same spot. C is then increased to the value of the next point on the c axis, by another FOR-NEXT loop, and the process repeated (line 500) until the whole range between C0 and C1 has been covered.

Listing 3

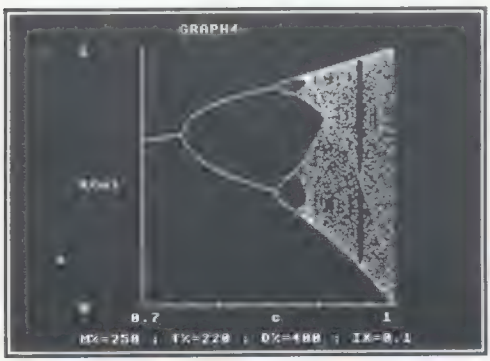
```
   10 REM Program CHAOS3
   20 REM Version B1.2
   30 REM Author  Jim Vernon
   40 REM BEEBUG  March 1990
   50 REM Program subject to copyright
   60 :
  100 ON ERROR:PROCerror:END
  110 MODE1
  120 PRINT'"   X(n+1)=4*c*X(n)*(1-X(n))
"
  130 PRINT'"This prog. graphs X(n) agai
nst"'"changing c. Ranges of X and of c a
nd"'"the initial value of X can be chose
n."
  140 PRINT'"The range of c is divided i
nto D%"'"intervals and X is iterated M%
times"'"for each value of c and those wi
thin"'"the defined range of X graphed."
  150 PRINT'"The total number of iterati
ons per"'"value of c (M%) and the number
 to be "'"ignored (T%) can also be varie
d."
  160 REM Input constants, ranges etc.
  170 INPUT'"Limits of c (<1) (=C0,C1)",
C0,C1
  180 INPUT"Limits of X (<1) (=X0,X1)",X
0,X1
  190 INPUT"Initial X (=IX)",IX
  200 INPUT"No. of points on C axis(D%)"
,D%
  210 INPUT "Total No. of iterations per
 val.c(=M%)",M%
  220 INPUT"No. not to be printed(T%)",T
%
  230 INPUT"NAME of File",A$
  240 CLS
  250 REM Draw axes (with 10 divisions),
 mark ranges etc.
  260 PROCaxis(200,150,200,950)
  270 FOR Q%=1 TO 10:PROCaxis(200,150+Q%
*80,210,150+Q%*80) :NEXT
  280 PROCaxis(200,150,1000,150)
  290 FOR Q%=1 TO 10:PROCaxis(200+Q%*80,
150,200+Q%*80,160) :NEXT
  300 PRINT TAB(10,0);A$
  310 PRINT TAB(0,2);X1:PRINT TAB(0,15)"
X(n)":PRINT TAB(0,27);X0
  320 PRINT TAB(5,28);C0:PRINT TAB(19,28
)"c":PRINT TAB(30,28);C1
  330 PRINT TAB(0,30)"M%=";M%;" : T%=";T
%;" : D%=";D%;" : IX=";IX
  340 REM Calculate values of c and reit
erated values of X(n)
  350 K=C1-C0:P=K/D% :L=X1-X0
  360 FOR C=C0 TO C1 STEP P
  370 X=IX
  380 FOR N% = 1 TO M%
  390 X=FNformula(X)
  400 REM First T values of X(n) not sho
wn
  410 IF N%<=T% GOTO 480
  420 REM From N%>T% start plotting
  430 Y%=(X-X0)*800/L+150
  440 R%=(C-C0)*800/K+200
  450 REM X(n) not plotted if outside ch
osen range
  460 IF Y%<150 OR Y%>950 GOTO 480
  470 PLOT 69,R%,Y%
  480 NEXT
  490 REM Same calculations and plot for
 next value of c
  500 NEXT
  510 OSCLI("SAVE "+A$+" 3000 8000")
  520 END
 1000 :
 1010 REM Formula defined
 1020 DEF FNformula(Z)=4*C*Z*(1-Z)
 1030 :
 1040 REM Procedure for drawing lines to
 make axes etc.
 1050 DEFPROCaxis(A%,B%,G%,H%)
 1060 MOVE A%,B%
 1070 DRAW G%,H%
 1080 ENDPROC
 1090 :
 1100 DEF PROCerror
 1110 VDU7:REPORT:PRINT" at line ";ERL
 1120 ENDPROC
```

Many interesting graphs can be drawn with this program. To start with, the effect can be examined of including the initial values of X(n) i.e. T% is made 0 or a small number. Figure 3 shows the result when M% = 8, T% = 0 and IX = 0.9. The picture is that of several polynomial functions fluctuating with increasing amplitude as c increases and the beginnings of a chaotic state when c is near to value 1. By experimenting with different values of IX and low values of M% and T%, a wide range of graphs is obtained, basically similar to figure 3 but with differing frequencies and amplitudes.



easing complexity as c increases in value

As M% is steadily increased, and with T% kept at 80% to 90% of M%, the fluctuations of figure 3 disappear and a different picture gradually emerges. Beyond M%=100 and T%=90 no further major change develops but there is greater clarity in some regions. At this level changing the initial value of X (IX) has no effect. Figure 4, with M%=250 and T%=220, is a typical example of the final picture. It gives a visual confirmation of the results already obtained from the program CHAOS1, with first a single line for X(n), then a succession of bifurcations starting at c = 0.75 and eventually chaos. As c increases, when bifurcations occur they are arranged vertically i.e. they occur at the same value of c whatever branch they are on.



For values of c>0.7 population changes appear to degenerate into chaos

The path of X(n) as it forks and forks again can be further studied by magnifying figure 4, using reduced values of the ranges X0,X1 and C0,C1 (X's limited range is only for the purposes of the graph, the iterative calculations still cover the full range of X from 0 to 1). Table 1 gives the values of C0,C1 and X0,X1 for insertion into the program CHAOS3 which will steadily magnify the nodes at the top of figure 4, until a magnification of 10000 times is reached. M% and T% have to be increased at these high magnifications to make the position of the nodes clear, so that even if the value of D is kept at 400, the program may take several hours. It is worth it, however, to see that no matter how high the magnification, bifurcation continues with the broad shape remaining the same.

C0 - C1	X0 - X1	IX	M	T
0 1	0 1	0.9	10	0
0.7 1	0 1	0.9	10	0
0.7 1	0 1	0.5	1000	950
0.8 0.9	0.8 0.9	0.5	500	475
0.89 0.8928	0.89 0.893	0.3	2000	1900
0.892 0.8926	0.892 0.8924	0.5	4000	3800
0.8924 0.8925	0.8924 0.8925	0.5	5000	4700

Table 1. Parameters for Magnifications

From printouts of these various graphs, the values of c at the nodes can be measured reasonably accurately. Table 2 gives measured values of c for the first 6 groups of nodes and shows how the gap between the values of c rapidly falls away. The decline is in fact geometric, with a dividing factor whose limit for a very large number of forks is 4.669... (Feigenbaum's constant). The figures in the last column of Table 2 are all close to this figure.

Value of c at node	Distance apart of nodes	1st. distance divided by 2nd.
0.75		
	0.11236	
0.86236		4.753
	0.02364	
0.8860		4.626
	0.00511	
0.89111		4.776
	0.00107	
0.89218		4.280
	0.00025	
0.89243		

Table 2. Nodes in 4c.X.(1-X)

There are even more interesting aspects to this program; however, space dictates that these will have to be left to next month. In the meantime, see what you can discover for yourself.

WORKS OF REFERENCE

CHAOS - Making a New Science by James Gleick, Sphere Books 1989 (see review in this issue).

Exploring the Geometry of Nature by Edward Reitman, Windcrest Books 1989.

Soft Function Keys Within View

Peter Vine explains how to incorporate use of the function keys with View and improve the efficiency of your word processing.

If a word or phrase is to be used often while editing with View, it is possible to program a function key with this string. This will not only save some time for us common two-fingered typists, but will also reduce the number of errors caused by trying to type the recurring phrase too quickly. This is especially true if the text being edited is a source listing for a computer program.

```
      HOW MANY CHAR. TO SEARCH IN INPUT RECORD     00001150
      WRITE(OUTPUT,23)                             00001160
   23 FORMAT("ENTER FILE TYPE")                    00001170
      READ(INPUT,24)X                              00001180
   24 FORMAT(A7)                                   00001190
```

The manual for View version 3.0 explains the procedure on page 112, but there is no mention in the View Guide or Into View for View 2.1, even though the technique works just as well there. The function keys are already used by View for its various commands, both by themselves, and with the Shift or Control keys pressed. The fourth combination, using both Shift and Control together, isn't used however, and so by issuing the OS command *FX228,1 from the command screen, pressing Ctrl-Shift-fkey will cause that key to be expanded as a soft key, and any programmed string to be entered into the text.

By way of an example, I was recently typing in a Fortran program, and the print statement is rather more complicated than in Basic. From the command screen the following lines were entered:

```
*FX228,1
*KEY5"WRITE(OUTPUT,"
```

so every time I needed a write statement, I merely pressed Ctrl-Shift-f5, followed by what I wanted to be printed.

This program had eight-digit line numbers (at the end of each line). For ease of entry, only the last four digits were typed in, as the preceding zeros could be added afterwards. The following extract from the program shows some of the line numbers preceded with their zeros:

However, entering the four zeros on every line was going to be very tedious. The zeros were therefore programmed into f5, but it was still tricky to release the Shift and Control keys and then move down to the next line and back to the beginning of the new line number. What was needed was the ability to program in not only text characters, but also cursor key movements. This is all clearly explained in the Wordwise manual, but not in that for View.

If "cursor editing" is turned off with *FX4,1, then the left cursor key gives a character whose is value 138. I made a guess that this might have the desired effect, but as this code is greater than 128, a rather complicated character sequence is needed to tell the operating system to put that byte into the soft key buffer. It is explained on page 142 of the BBC micro User Guide (Chapter 25 - Programming the Red User Defined Keys), and on page 16 of the Advanced User Guide, that !! used in a *KEY definition adds 128 to the value of the following byte (where '!' is the character to the left of the cursor keys obtained by pressing Shift and '\' together). The value wanted was 138, i.e. 128 + 10, so "!!J" is required to program code 138 into a function key (where '!J' represents Ctrl-J which has code 10).

But it didn't work as expected - that code in fact simulated pressing the Delete key! After some trial (and a lot of error), it transpired that "!!X" would give the desired effect, and that "!!Z" gave the cursor-down command. Thus to get the desired effect function key f5 needed to be programmed as follows:

```
*KEY5"0000!!X!!X!!X!!X!!Z"
```

after which, positioning the cursor on the beginning of the first line number, ensuring insert mode was on (Ctrl-f4), and pressing Ctrl-Shift-f5 repeatedly, caused the zeros to be added as required.

While searching for the appropriate sequence, it became apparent that all of the function key operations of View, and those of the cursor and other keys, could be simulated by the same method. Table 1 shows the codes and key sequences needed to obtain each effect (codes 128 to 135, and 188 to 255 seem to have no function), from which it can be seen that "!!(" would give the equivalent to Shift-Cursor-Left (go back to start of word), which would be easier than all the backspaces used above.

This means that quite sophisticated editing strings can be pre-programmed into the ten function keys, to be invoked subsequently by pressing Shift-Ctrl in conjunction with the appropriate function key. Furthermore, a group of common function key definitions can be written as a Basic program, to define the function keys before even entering View. The ultimate in automation is to set up a !Boot file on your View disc which automatically chains this program before entering View itself. For example:

```
*BASIC
CHAIN"KeyDefs"
*View
```

Code	Sequence	Key	Effect
136	!H	Escape	Go to command screen
137	!I	Return	Go to start of new line
138	!J	Delete	Delete character before cursor
139	!K	Tab	Insert TAB code
(140)	!!L	(f0)	Format block
(141)	!!M	(f1)	Go to top of text
(142)	!!N	(f2)	Go to bottom of text
143	!O	f3	Delete end of line
144	!P	f4	To start of line
(145)	!!Q	(f5)	To end of line
(146)	!!R	(f6)	Insert line
147	!S	f7	Delete line
148	!T	f8	Insert character
149	!U	f9	Delete character at cursor
(150)	!!V	(f10)	No effect
151	!W	Copy	Copy selected text
152	!X	Left-cursor	Cursor Back
153	!Y	Right cursor	Cursor Forward
154	!Z	Down cursor	Cursor Down
(155)	!![	Up-cursor	Cursor Up
156	!\	Shift f0	Move
(157)	!!]	(Shift-f1)	Swap case
(158)	!!^	(Shift-f2)	Release Margins
159	!	Shift f3	Delete to character
160	!_	Shift f4	Highlight 1
161	!!	Shift f5	Highlight 2
(162)	!!"	(Shift-f6)	Go to Marker
(163)	!!#	(Shift-f7)	Set Marker
(164)	!!$	(Shift-f8)	Edit command
(165)	!!%	(Shift-f9)	Delete command
166	!&	Shift f10	No effect
167	!'	Shift Copy	Insert current ruler
168	!(	Shift left	Back to start of word
169	!)	Shift right	Forward to start of next word
170	!*	Shift down	Down a screenful
171	!+	Shift up	Up a screenful
(172)	!!,	(Ctrl-f0)	Delete block
173	!-	Ctrl f1	Next match
174	!.	Ctrl f2	Format mode
(175)	!!	(Ctrl-f3)	Justify mode
(176)	!!0	(Ctrl-f4)	Insert mode
177	!1	Ctrl f5	Default ruler
(178)	!!2	(Ctrl-f6)	Split line
179	!3	Ctrl f7	Concatenate lines
(180)	!!4	(Ctrl-f8)	Mark as ruler
(181)	!!5	(Ctrl-f9)	(No effect)
182	!6	Ctrl f10	(No effect)
183	!7	Ctrl-Copy	(No effect)
184	!8	Ctrl-left	Go to beginning of line
(185)	!!9	(Ctrl-right)	Go to end of line
(186)	!!:	(Ctrl-down)	Go to bottom of text
(187)	!!;	(Ctrl-up)	Go to top of text

Table 1. Key sequences for soft key definitions

For a more extensive discussion of the possibilities of using the soft keys with View, readers are referred to the two part article on this subject entitled Getting a Better View in BEEBUG Vol.4 Nos.9 & 10.

Dichotomous Keys (Part 2)

Rupert Thompson concludes the description of his expert system

Last month we introduced a program to edit and run simple dichotomous keys. The routines we described are fine as they stand, but it would be nice to have some more powerful editing techniques. These are provided by this month's routines. Type in and save the listing carefully, then merge it with last month's program using the following sequence of commands:

```
*SPOOL Dich2
LIST
*SPOOL
LOAD"Dich1"
*EXEC Dich2
SAVE"Dich"
```

or replace "Dich" in the last line with your own filename. No amendments need to be made to last month's listing.

QUESTION CHAINS

By virtue of the dichotomous structure, each question may point to two others below it in the key. Thus a chain of questions soon builds up. However, the simple editor presented last month merely treated each question individually, not as part of a chain. With this month's listing in place the editor automatically chains the questions to one another, a process which is totally invisible to the user. Consider the following arrangement, for example:

Question 1 (Root Level) points to
Question 2 and Question 3
Question 2 (Up to Question 1)
Question 3 (Up to Question 1)
Question 4 (Root Level)

When question 1 is edited, questions 2 and 3 are automatically connected to it in the chain. Now when you are in edit mode, as well as the question you are editing, the next level up in the chain is displayed above it on the screen. This makes constructing a key considerably easier.

A question is designated *Root Level* if it is not connected to an earlier question, otherwise the number of the earlier, or parent, question is given.

The *Up Lev* option moves back one step up the chain.

ROUTING

The techniques available for routing are very powerful. The simplest are connected with chaining.

Suppose that in the above example you re-define question 1 to point to 3 and 4, omitting 2 from the chain. Question 4 will be connected up automatically, but question 2 still thinks that it too is part of the chain. In fact it is not, and question 1 will never call it, but when editing question 2, question 1 will still be displayed above it on the screen, and *Up Lev* from 2 will go to 1. To remove 2 from the chain, select *Go To*, choose question 2, select *Routing* and then *Connect*. Since no other question points to 2, it will be removed from the chain. However, had it been pointed to by another question, but was still not part of the chain, then *Connect* would hook it in.

Since any keys designed with last month's listing will not be chained up, the effect of *Connect* can easily be demonstrated by selecting it for questions 2 or 3, for example. However, the easiest way to chain a whole key is with the *Re-Chain* option. This will attempt to connect each question in turn to the question which points to it. This is very useful if the chain becomes corrupted in some way.

RE-ROUTING

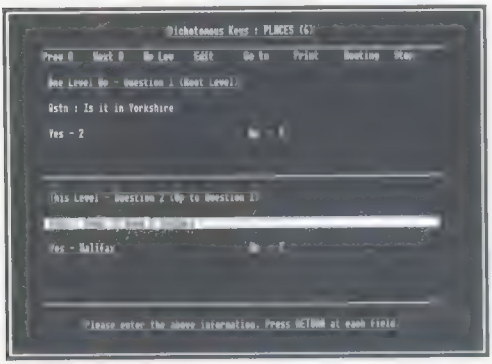
Suppose that you want to move a question from one position in a key to another. You could do this manually, but it would be very tedious. Instead, select *Routing* and then *Re-Route* to perform the transplant automatically. For example:

Question 1 points to 2 and 3
Question 2 points to 5 only
Question 3 points to 6 and 7
Question 4 (un-chained)

could be transformed to:

Question 1 points to 4 and 3
Question 2 (un-chained)
Question 3 points to 6 and 7
Question 4 points to 5 only

Thus question 2 has been moved to position 4, and the chain re-connected automatically. Be warned, though, that re-routing question 1 will prevent the correct operation of the key.



The editing screen showing the new functions

This technique can be used to swap the positions of questions in the key. You may remember from last month that one spare question was created when the key was set up. This can be used to swap two questions without erasing one of them. For example, consider a file containing 20 items. Only 19 questions are needed, leaving one, number 20 say, free. The following sequence will swap questions 3 and 7:

(i) re-route question 3 to question 20
(ii) re-route question 7 to question 3
(iii) re-route question 20 to question 7

Thus the unused question acts as a temporary store. You will notice that once again the whole chain has been re-connected.

PRINTING

The final utility this month is a routine to print out the key onto paper. Simply select *Print* for a paper copy in a similar format to that shown at the start of Part 1.

USING DICHOTOMOUS KEYS

Finally, it will be helpful to discuss some ideas on the use of keys. They are very easy to use, even for non-experts. The program presented here is a type of Expert System - it has a knowledge base from which ordinary people can identify anything known to the system. Thus the program has a practical use in addition to providing a demonstration of Artificial Intelligence. For example, you could let a group of pupils identify chemical compounds for themselves with the help of a key. Alternatively, a medical knowledge base could be used for self-diagnosis. This is in fact being tried in some surgeries, although the knowledge base needs to be very large, and the doctor must only use the diagnosis as a second, not always entirely reliable, opinion. This is not to suggest that the key will get it wrong, but that the patient might well give a wrong answer, and a life could be at stake!

Bear in mind, of course, that a key can only identify something it knows - if you need proof of this, try to identify a circle with the key shown in Part 1, and see what result you get!

```
 2200 IF Z=2 AND ptr>0 THEN qstn=ptr
 2230 IF Z=5 THEN PROCprintkey
 2240 IF Z=6 THEN PROCroute
 2830 PROCchain
 2850 DEF PROCchain:IF VALyes$>0 THEN I=
VALyes$:PROCreadrec(I):ptr=qstn:PROCwrit
erec(I)
 2855 PROCreadrec(qstn):IF VALno$>0 THEN
 I=VALno$:PRCreadrec(I):ptr=qstn:PROCwr
iterec(I)
 2860 ENDPROC
 2865 DEF PROCconnect
 2870 LOCAL I,n:I=0:n=0
 2880 REPEAT:I=I+1:PROCreadrec(I):IF VAL
yes$=qstn OR VALno$=qstn THEN n=-1
 2890 UNTIL I=ite% OR n:IF NOT n THEN I=
0
 2900 PROCreadrec(qstn):ptr=I
 2910 ENDPROC
 2920 DEF PROCroute
 2930 LOCAL Z
 2940 lotus%=-1
 2950 Z=FNlotus("Re-Chain*Connect*Re-Rou
te*",2,-1,-1,0)
```

```
 2960 IF Z=0 THEN PROCrechain
 2965 IF Z=1 THEN PROCconnect:PROCwriter
ec(qstn)
 2970 IF Z=2 THEN PROCreroute
 2980 ENDPROC
 3180 DEF PROCprintkey
 3190 IF NOT FNsure("Print out Key.") TH
EN ENDPROC
 3200 LOCAL I
 3210 PROCwindow(3,-1):VDU2
 3220 PROCcent("Dichotomous Keys : "+fil
e$)
 3230 PROCcent("("+STR$ite%+" Items)")
 3240 PRINT:FOR I=1 TO ite%
 3250 PROCreadrec(I):IF ptr>0 OR I=1 THE
N PROCentry
 3260 NEXTI
 3270 VDU3:CLS:ENDPROC
 3280 DEF PROCentry:PRINT
 3290 PRINTTAB(5);I;TAB(9);q$
 3300 PRINTTAB(9);"If YES then ";FNifyes
(yes$)
 3310 PRINTTAB(9);"If NO  then ";FNifyes
(no$)
 3320 ENDPROC
 3330 DEFFNifyes(o$):IF VALo$=0 THEN ="i
t is "+o$ ELSE ="go to Question "+o$
 3340 DEF PROCcent(o$):PRINTTAB((80-LENo
$)/2);o$:ENDPROC
 3420 DEF PROCreroute
 3430 LOCAL I
 3440 PROCwindow(6,-1):I=VALFNenter(22,0
,3,"","0123456789","Move this Question (
"+STR$qstn+") to Question"):CLS
 3450 IF I=0 OR I>ite% THEN CLS:ENDPROC
 3460 IF ptr>0 THEN p=ptr:PROCreadrec(pt
r):IF VALyes$=qstn THEN yes$=STR$I ELSE
no$=STR$I
 3465 IF p>0 THEN PROCwriterec(p)
 3470 PROCreadrec(qstn):PROCwriterec(I)
 3480 ptr=0:q$="":yes$="":no$="":PROCwri
terec(qstn):qstn=I
 3490 PROCreadrec(qstn):I=VALyes$:IF I>0
 THEN PROCreadrec(I):ptr=qstn:PROCwriter
ec(I)
 3500 PROCreadrec(qstn):I=VALno$:IF I>0
THEN PROCreadrec(I):ptr=qstn:PROCwritere
c(I)
 3510 ENDPROC
 3520 DEF PROCrechain:q=qstn
 3530 PROCwindow(3,-1):PRINTTAB(30,8);"C
haining Question ";:FOR qstn=2 TO ite%:P
RINTTAB(48,8);qstn:PROCreadrec(qstn):IF
q$<>""AND yes$<>""AND no$<>""THEN PROCco
nnect:PROCwriterec(qstn)
 3540 NEXTqstn:qstn=q:ENDPROC
```

Upgrading to an Archimedes (continued from page 8)

an analogue RGB connection (to obtain up to 256 colours on the screen at any one time). The Beeb uses a TTL system via DIN connectors. However, if you have a monochrome monitor which works from the BNC or UHF sockets, then this will work just as well with the Archimedes, though you will miss the pleasure of all those extra colours of course. If you decide to keep your Beeb, you can usually use an Arc colour monitor with it with a suitable lead.

If you are buying a new colour monitor for an Archimedes then you have a choice between the standard colour monitor and costing about £220, or a so-called multi-sync monitor, which doubles the horizontal resolution allowing several additional high resolution screen modes to be used, but at a cost of some £520.

You should have realised by now that any sideways ROM or RAM boards, shadow RAM boards and similar have no place on an Archimedes (though ROM-based software may still be usable). Likewise, teletext adaptors and second processors cannot be carried forward to the new system. However, any modem which you have may well be usable with an Archimedes, though you will need a new connecting cable. The controlling software may still be usable (but see *Software Compatibility* next month).

If you use any equipment which connects to the user port, analogue port or 1MHz bus, then you may still be able to use that (by buying an I/O podule), but there may be a problem with any controlling software, more than likely if it is ROM based. You will need to check this area individually.

Next month I will conclude this article by considering the important question of software compatibility, as well as looking at the promised case studies, and a comprehensive summary of the whole question of upgrading.

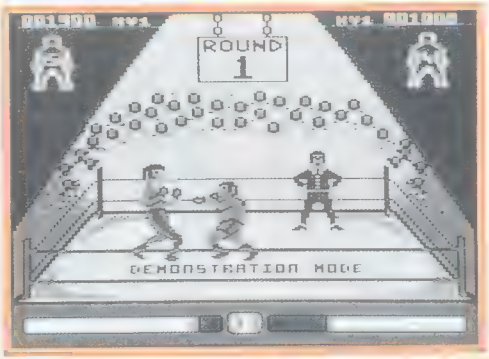
Games Review

Products: Play It Again Sam 12
The Last Ninja 2

Supplier: Superior Software
P.O.Box 6, Brigg,
South Humberside DN20 9NH.
Tel. (0652) 58585

Prices: £9.95 (BBC & Electron cassette)
£11.95 (5.25" disc)
£14.95 (3.5" disc for the Compact)
(Available from BEEBUG - all prices inc. VAT - p&p extra)

Yet another Play It Again Sam! We are now up to number 12 in this apparently never-ending series of four-game compilations from Superior. This one features two oldies, *Skirmish* and *Blagger*, plus two fairly recent releases, *By Fair Means or Foul* and *The Last Ninja*. The inclusion of the latter two on this compilation is surprising as they are really not that old.



By Fair Means or Foul

The Last Ninja is a game based on the theme of martial arts, where you guide your man in his quest to reach the inner sanctum of the palace of Lin Fen. You must collect objects and weapons along the way to help you achieve your goal. As usual, there are many hazards to contend with, puzzles to solve, and assailants to overcome. The game features some nice graphics with good animation. However, game play can be a bit of a trial at first with the number of key combinations to learn and remember.

By Fair Means or Foul is a boxing game that can be played by one or two players. The general idea of this game is to knock hell's bells out of your opponent. There are different legitimate punches and moves you can make, but if they fail you can always throw in the odd dodgy one when the referee isn't looking. This game is again not easy to master, needing the use of some nine keys. A joystick, to my mind, is a 'must' unless you are blessed with the manual dexterity of Fagin.

Next we have *Skirmish*. This is a version of the old arcade classic 'Joust'. The game consists of guiding a knight around the screen on the back of a large bird, while being attacked by other such birds. To add to your troubles there are a few Pterodactyls also intent on doing you harm. It is quite a simple game with rather sparse graphics, but nevertheless, very addictive.

Lastly, we have *Blagger*. This is a real oldie which was originally released by Alligata

Blagger

way back in the Beeb's early days. It is a platforms and levels game where the object is

Continued on page 56

Using the ROM Filing System (Part 2)

An illustrative completion to the explanation of the ROM Filing System by describing a program to load files to the RFS.

Following last month's program, which created the ROM header for the ROM Filing System, this month's program, RFSload, will allow you to load any suitable machine code and Basic programs, or EXEC files, into RFS memory (sideways RAM) using the ROM header.

The new program should be typed in and saved. Before attempting to use RFSload, make sure that you have already installed the RFS ROM image header, produced by last month's program, in a suitable RAM bank and pressed Ctrl-Break. Once a suitable RFS header, or an already partially full RFS, has been installed (you can always add additional files to an existing RFS system), RFSload will then load one or more selected files in turn into memory at &4000. It formats each file with the necessary header blocks and CRC data, as it is written to sideways RAM in the RFS format.

The format the RFS uses is exactly the same as the CFS (Cassette Filing System) format. The file data is split into blocks of 256 bytes. The first and last blocks of each file are formatted with a header block, which contains the file information, as well as the Cyclic Redundancy Check (CRC) data for the current block. For further details about the RFS refer to the User Guides or The BBC Micro ROM Book.

PROGRAM INFORMATION

When run, the program first prompts for the RFS Title as shown by *ROMS and the RAM bank number. This information is then checked to ensure that the wrong RAM image is not written to. After this PROCsrl requests the start address in sideways RAM at which to load the files. This is marked by an end-of-RFS marker byte (&2B). If this address is not known, entering an invalid address, outside the range &8100 to &BFD0, will simply cause PROCgetmark to catalogue the RFS and find the marker for itself.

When the position of the end-of-RFS marker has been verified, a REPEAT-UNTIL loop then calls PROCarrange until there are no more files to be loaded. This procedure checks the filing system in use, and sets the input verification for the "Load File ...." prompt. When this prompt appears, it is possible to use most OS and filing system commands to select, change or catalogue the directory.

When the "Any more Files to Load" prompt is answered with 'N', the program terminates and gives the necessary information to save the RAM image to disc. All that now remains is to press Ctrl-Break to let the OS know that it has a ROM Filing System active. Type *ROM and use *EX, or *CAT with *OPT1,2 to check the file information. The RFS image can always be loaded back into sideways RAM at a later date, and the files it contains accessed using the documented RFS commands (*CAT etc), once the RFS has been selected with the *ROM command.

The program will run in mode 7 with PAGE at &1900 on a model B, provided that sideways RAM has been fitted. However, it may be necessary for you to alter each occurrence of the Master's *SRWRITE command (contained within OSCLI statements) to suit your type of sideways RAM (refer to the instructions supplied with this). Also the character validation in FNverchr("4567",1) at line 2060, will need to be changed if the RAM banks are not labelled 4, 5, 6 and 7. The correct numbers for the RAM banks should be inserted in their place.

TECHNICAL INFORMATION

For the sake of ease and speed I have used assembler, not only for Cyclic Redundancy Checking and RFS search procedures, but also for the OSFILE parameter block and RFS header block procedures, with one pass and the OPT variable set to zero.

The program is well error trapped by the relevant messages and also PROCerrs. This is to ensure that any wrong commands that might be given or filing system clashes are recoverable. The 'Y/N' prompt can usually be answered 'Yes', unless there is a fault in the program listing itself.

HINTS ON USE

Owing to the slow access speed of the RFS, it seems pointless to put long programs in the RFS, although a file can spread across any two consecutive RFS banks as long as the block numbers are concurrent. Personally I tend to use it for printer, disc and debugging routines that are usually needed when a different filing system is in use from the one which contains them.

As written, RFSload is best used if the files you wish to write to the RFS are in the current directory. This then allows the maximum number of characters for the file name, and it will be loaded into the RFS with this name.

When the command *-ROM-LIBFS is used on a Master, the RFS will be searched for a file automatically, if it is not on the current filing system. Also, it can be used in a temporary manner by using *-ROM-RUN <fsp>.

If a machine code program called !BOOT is written and loaded into the RFS, then when *ROM followed by Shift-Break is used, the RFS will automatically *RUN the !BOOT file (just as with any other filing system).

Remember that when any files are run or loaded from the RFS, it will remain "Paged In" as the current filing system, so if your own program requires DISC or other ROM access then the relevant OS commands will have to be inserted at the start of the program before loading it into the RFS in the first place (with RFSload).

```
 10 REM  Program RFSload
 20 REM  Author  Jon Keates
 30 REM  Version B3.03
 40 REM  BEEBUG  March 1990
 50 REM  Program subject to copyright
 60 :
100 MODE135
110 HIMEM=&3FFF
120 @%=4:VDU23,1,0;0;0;0;
130 osfile%=&FFDD:osrdrm%=&FFB9
140 oswrch%=&FFEE:osbyte%=&FFF4
150 osnewl%=&FFE7:osargs%=&FFDA
160 fload%=&4000:crc%=&900:smark%=&900
170 hblok%=&930:fblok%=&960
180 fname%=&980:footer%=&990
190 valid$="":FOR A%=32 TO 126
200 valid$=valid$+CHR$(A%):NEXT
210 ON ERROR PROCerrs
220 REPEAT PROCinfo
230 trom$="":?&F6=&09:?&F7=&80:Y%=ramb
lk%
240 REPEAT C%=(USR(osrdrm%) AND &FF)
250 IF C%<>0 trom$=trom$+CHR$(C%)
260 ?&F6=?&F6+1:UNTIL C%=0 OR ?&F6=&14
270 IF trom$="" OR trom$=CHR$255 trom$
="NOTHING"
280 IF title$<>trom$ PROCmess(0,trom$+
" is in RAM bank "+STR$~ramblk%) ELSE PR
OCmess(-1," ...  Verified.")
290 UNTIL flag%=TRUE
300 PRINTTAB(2,6)"RAM bank : "CHR$130;
ramblk%;
310 PRINTTAB(16,6)CHR$135"RFS .. : "CH
R$130;title$
320 REPEAT PROCsrl
330 UNTIL flag%=TRUE
340 VDU26:mleft%=&BFFF-srload%
350 ON ERROR PROCerrs
360 PROCchek
370 REPEAT flag%=TRUE
380 PRINTTAB(28,9)~mleft%
390 PRINTTAB(3,23)"  End of RFS maker
at  "CHR$129;"&";~srload%;SPC5
400 PROCarrange
410 mleft%=&BFFF-srload%:PRINTTAB(28,9
)~mleft%
420 PRINTTAB(3,23)CHR$132"Any more Fil
es to load .. Y/N :";
430 REPEAT ans$=GET$:UNTIL INSTR("YyNn
",ans$)>0
440 UNTIL INSTR("Nn",ans$)>0
450 VDU28,0,22,39,12,12
460 PRINT'"To Save RFS Image use   ..
."
470 PRINT'"*SRSAVE ";title$;" 8000 ";
~srload%+1;" ";~ramblk%
480 END
490 :
1000 DEF PROCarrange
1010 A%=0:Y%=0:X%=&80
1020 IF (USR(osargs%) AND &FF)=4 lmt%=7
ELSE lmt%=10
1030 VDU28,0,22,39,12,12
1040 PRINT''SPC6;"Load File  ? ..."
1050 PRINT''CHR$133SPC6;:file$=FNverchr
(valid$,lmt%)
1060 IF LEFT$(file$,1)="*" PROCoscom:EN
DPROC
1070 PROCloadfile
1080 IF flag%=FALSE VDU26:ENDPROC
1090 PRINT'"    File info  :  ";~(flen
%DIV256);"  ";~flen%
1100 ptr%=fload%:bnos%=0
1110 REPEAT
1120 IF flen%<257 THEN status%=&80:mark
```

```
er%=&2B:blen%=flen% ELSE status%=&00:mar
ker%=&23:blen%=256
 1130 IF status%=&80 OR bnos%=0 PROChead
block ELSE srload%=srload%+1
 1140 OSCLI"SRWRITE "+STR$~ptr%+" +"+STR
$~(blen%)+" "+STR$~srload%+" "+STR$~ramb
lk%
 1150 srload%=srload%+blen%
 1160 ?&8B=ptr%:?&8C=(ptr% DIV 256):?&8F
=blen%
 1170 CALLcrc%
 1180 ?footer%=?&8E:footer%?1=?&8D:foote
r%?2=marker%
 1190 OSCLI"SRWRITE "+STR$~footer%+" +3
"+STR$~srload%+" "+STR$~ramblk%
 1200 srload%=srload%+2:flen%=flen%-blen
%
 1210 bnos%=bnos%+1:ptr%=ptr%+blen%
 1220 UNTIL status%=&80:SOUND1,-12,160,6
:VDU26
 1230 ENDPROC
 1240 :
 1250 DEF PROCloadfile
 1260 PROCparablok
 1270 A%=5:X%=fblok%:Y%=(fblok% DIV 256)
 1280 IF (USR(osfile%) AND &FF)<>1 PROCm
ess(0,"  File not found .. "):ENDPROC
 1290 ladr%=fblok%!2
 1300 xadr%=fblok%!6
 1310 flen%=fblok%!10
 1320 eof%=FNlastbyte
 1330 IF eof%>&BFFF PROCmess(0,"file too
big for RFS RAM ..."):ENDPROC
 1340 PROCparablok
 1350 A%=&FF:X%=fblok%:Y%=(fblok% DIV 25
6)
 1360 CALLosfile%
 1370 ENDPROC
 1380 :
 1390 DEF PROCheadblock
 1400 P%=hblok%
 1410 [ OPT 0
 1420 EQUB &2A:EQUS file$:EQUB &0
 1430 EQUD ladr%:EQUD xadr%
 1440 EQUW bnos%:EQUW blen%
 1450 EQUB status%:EQUD eof%
 1460 ]
 1470 ?&8B=hblok%+1:?&8C=(hblok% DIV 256
)
 1480 ?&8F=(LEN(file$)+18)
 1490 CALLcrc%
 1500 [ OPT 0
 1510 EQUB ?&8E:EQUB ?&8D
 1520 ]
 1530 OSCLI"SRWRITE "+STR$~hblok%+" "+ST
R$~P%+" "+STR$~srload%+" "+STR$~ramblk%
 1540 srload%=srload%+(P%-hblok%)
 1550 ENDPROC
 1560 :
 1570 DEF PROCparablok
 1580 $fname%=file$
 1590 P%=fblok%
 1600 [ OPT 0
 1610 EQUW fname%:EQUD fload%
 1620 EQUD 0:EQUD 0:EQUD 0
 1630 ]
 1640 ENDPROC
 1650 :
 1660 DEF FNlastbyte
 1670 IF flen%>256 sets%=2 ELSE sets%=1
 1680 IF flen%>512 ext%=((flen% DIV 256
+(flen% MOD 256=0))-1 ELSE ext%=0
 1690 =srload%+((LEN(file$)+23)*sets%)+f
len%+ext%*3
 1700 :
 1710 DEF PROCmess(er%,prt$)
 1720 VDU7,12
 1730 PRINT''CHR$131"    ";prt$
 1740 PROCwait(2):VDU26
 1750 flag%=er%:ENDPROC
 1760 :
 1770 DEF FNverchr(test$,num%)
 1780 PRINT SPC(num%);CHR$124;
 1790 FOR a%=0 TO num%:VDU8:NEXT
 1800 LOCAL G%,got$:got$="":*FX15
 1810 REPEAT G%=GET
 1820 IF G%=127 AND LEN(got$)>0 VDU8,32,
8:got$=LEFT$(got$,LEN(got$)-1)
 1830 IF LEN(got$)=num% G%=13
 1840 IF INSTR(test$,CHR$(G%)) got$=got$
+CHR$(G%):VDU G%
 1850 UNTIL G%=13 AND LEN(got$)>0
 1860 =got$
 1870 :
 1880 DEF PROCwait(t%)
 1890 T%=TIME:REPEAT UNTIL TIME>T%+100*t
%
 1900 ENDPROC
 1910 :
 1920 DEF PROCoscom
 1930 VDU12,14
 1940 requ$=RIGHT$(file$,LEN(file$)-1)
 1950 OSCLI requ$
 1960 PRINT'"   Press any Key to contin
ue."
 1970 G=GET:VDU12,15,26
 1980 ENDPROC
 1990 :
 2000 DEF PROCinfo
 2010 CLS:PRINT
 2020 PRINTCHR$141CHR$134SPC7"RFS  RAM
LOADER"
```

```
 2030 PRINTCHR$141CHR$134SPC7"RFS  RAM
LOADER"
 2040 VDU 28,0,22,39,12,12
 2050 PRINT''SPC6"RAM bank ? .. ";
 2060 rm$=FNverchr("4567",1)
 2070 ramblk%=VAL(rm$)
 2080 VDU12
 2090 PRINT''SPC6"RFS  name ? .. ";
 2100 title$=FNverchr(valid$,10)
 2110 ENDPROC
 2120 :
 2130 DEF PROCsrl
 2140 PRINTTAB(3,9)"Available RFS Memory
 : "CHR$129"&3FFF"
 2150 VDU 28,0,22,39,12,12
 2160 PRINT''SPC6"Sideways RAM start add
ress .."
 2170 PRINT'SPC7"or enter FF for search.
"
 2180 PRINT''CHR$133SPC8"&";
 2190 side$=FNverchr("0123456789ABCDEF",
4)
 2200 srload%=EVAL("&"+side$)
 2210 IF srload%<&8100 OR srload%>&BFD0
VDU7:PROCgetmark
 2220 ?&F6=srload%:?&F7=(srload% DIV 256
):Y%=ramblk%
 2230 IF (USR(osrdrm%) AND &FF)<>43 PROC
mess(0,"End of RFS not found ...") ELSE
PROCmess(-1,"....  Verified at &"+STR$~s
rload%)
 2240 ENDPROC
 2250 :
 2260 DEF PROCgetmark VDU12
 2270 PRINT''CHR$131SPC6"Invalid SRAM ad
dress."
 2280 PRINT''CHR$131SPC9"Seaching  ... "
 2290 FOR pas=0 TO 1
 2300 P%=smark%
 2310 [ OPT pas*2
 2320 LDX #&D:LDY #(ramblk% EOR &F)
 2330 LDA #&8F:JSR osbyte%
 2340 LDA &F6:STA &70:LDA &F7:STA &71
 2350 CPX #0:BEQ enter
 2360 BRK:EQUS " ... RFS not found":BRK
 2370 .newl
 2380 JSR osnewl%:LDX #6:LDA #32
 2390 .gap
 2400 JSR oswrch%:DEX:BPL gap
 2410 .loop
 2420 JSR getbyte:CMP #&0:BEQ endn
 2430 JSR oswrch%:JMP loop
 2440 .endn
 2450 CLC:LDA &70:ADC #13:STA &70
 2460 LDA &71:ADC #&0:STA &71
 2470 JSR getbyte:PHA
 2480 JSR getbyte:STA &71
 2490 PLA:STA &70
 2500 .enter
 2510 JSR getbyte
 2520 CMP #&2A:BEQ newl
 2530 CMP #&2B:BEQ mark
 2540 BRK:EQUS " ... BAD RFS":BRK
 2550 .mark
 2560 DEC &70:RTS
 2570 .getbyte
 2580 LDA &70:STA &F6:LDA &71:STA &F7
 2590 LDY #(ramblk% EOR &F)
 2600 LDX #&E:LDA #&8F:JSR osbyte%
 2610 LDA &F6:STA &70:LDA &F7:STA &71
 2620 TYA:RTS
 2630 ]:NEXT
 2640 CALLsmark%
 2650 srload%=?&70+?&71*256
 2660 IF ramblk%<>(?&F5 EOR &F) PRINT''C
HR$129"    WRONG RAM Bank  !":VDU7
 2670 PRINT''CHR$130"   End of RFS at &"
;~srload%;" RAM Bank ";(?&F5 EOR &F)
 2680 PROCwait(3)
 2690 ENDPROC
 2700 :
 2710 DEF PROCchek
 2720 FOR pas=0 TO 1
 2730 P%=crc%
 2740 [OPT pas*2
 2750 LDY #0:STY &8E:STY &8D:CLC
 2760 .loop1
 2770 LDA &8E:EOR (&8B),Y
 2780 STA &8E:LDX #8
 2790 .loop2
 2800 LDA &8E:ROL A:BCC skip
 2810 LDA &8E:EOR #8:STA &8E
 2820 LDA &8D:EOR #16:STA &8D
 2830 .skip
 2840 ROL &8D:ROL &8E:DEX
 2850 BNE loop2:INY
 2860 CPY &8F:BNE loop1
 2870 RTS
 2880 ]
 2890 NEXT
 2900 ENDPROC
 2910 :
 2920 DEF PROCerrs
 2930 REPORT
 2940 IF ERR<>0 PRINT'"   Error ";ERR;"
 at line ";ERL ELSE END
 2950 PROCwait(1):*FX15
 2960 PRINT'"   Continue   ....   Y/N  ?
"
 2970 G=GET:IF (G AND &DF)=ASC"Y" ENDPRO
C ELSE STOP
 2980 ENDPROC
```

Statistics for Pleasure

by themselves or linked together.

Statistics can be fun - as this program is meant to be - offering insight into the fluctuations of repeated readings in science or of specified values in technology. It demonstrates *Gaussian distributions* (also known as *normal distributions*). An example is the distribution of IQ levels on either side of the mean throughout the population.

In the program listed here, up to three variables each with a normal distribution can be given independent values for mean and standard deviation. The program then shows how any chosen function of those variables fluctuates. It offers values for the mean and deviation of the function and also displays - crawling up the screen - histograms of the selected variables and of the specified function.

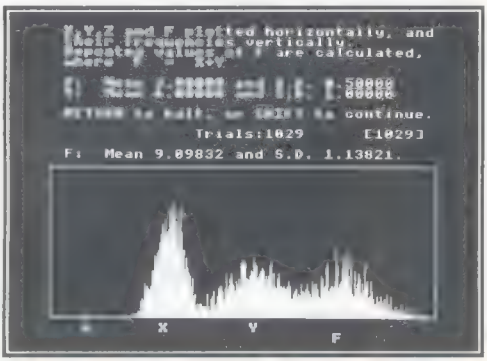
The program can be used in two ways: firstly, to give an insight into the way that statistical variables fluctuate and how they can be shown on histograms; and secondly, for study of the (rather difficult) way in which the standard deviation of a function of statistical variables depends on the standard deviations of the variables themselves.

The program is in pure Basic, so you only need to type it in, save it, then run it. You can put it to work straight away on the following example, which is the easiest way to explain what the program does.

USING THE PROGRAM

Suppose that a rod is made of two parts, one 40mm long and the other 60mm long. Of course the total length of the rod is 100mm. But now suppose that in manufacture the lengths fluctuate, the first part with a standard deviation of 3mm and the second with a standard deviation of 4mm. Suppose that the varying lengths follow a normal (Gaussian) distribution. What will be the deviation of the combined rod? A manufacturer who expected it to be 7mm would be unduly pessimistic!

What the program will do, given this example, is to display histograms that grow as the manufacture of the rods is simulated. Without further mathematics, the standard deviation of the lengths of the combined rods can be found.



Using the program to display a histogram and statistical information

When asked for a function, key in X+Y (i.e. the total length of the composite rods), and then the means and standard deviations of X as 40, 3 and Y as 60, 4. There is no need to specify the number of trials when asked - just press Return. As the program runs, the mean will naturally approach 40 + 60 = 100mm. We can also see what happens to the standard deviation. Pressing Return again halts the program to display the up-to-date values of the mean and the standard deviation. Pressing Shift allows the program to continue. By tapping Shift while the program is running you can 'capture' the values of mean and standard deviation at any moment without halting the operation. Holding down Shift displays the mean and standard deviation continuously - at the cost of slowing the program down. You can also single-step the program by holding down Return and tapping Shift.

There is a traditional divide between mathematics, a *priori* and true of all possible worlds, and science, tested empirically and applying to our own world as it is. What we have in this program, however, is empirical maths! At any rate, I hope readers enjoy watching the histograms grow and experimenting with different functions.

PROGRAM OUTLINE

PROCVariables allows the function and the means and standard deviations of its variables to be input, and then uses dummy data to test that EVAL will accept the function. PROCInitialValues and PROCPicture set up the histograms with suitable scaling.

At the start of the main REPEAT loop, PROCfind calls FNGauss, which repeatedly uses RND to churn out values of a variable which is almost normally distributed with mean of zero and standard deviation of unity. Perhaps readers can offer a faster way of doing this and speed the whole program up - for those of us without an Archimedes!

PROCFind uses the values from FNGauss to produce random variables X, Y and Z with the specified means and standard deviations and calculates from them values of the function F. PROCdisplay slots values of X, Y, Z and F into arrays for the histograms. PROCStat works out and displays the mean and standard deviation of F.

```
 10 REM Program Gauss61
 20 REM Version B1.0
 30 REM Author  Michael Taylor
 40 REM BEEBUG  March 1990
 50 REM Program subject to copyright
 60 :
100 @%=&02050A:Vscl=8:Read%=0
130 PP=RND(-TIME):MODE4
160 VDU19,0,7,0;0;0;
165 VDU19,1,0,0;0;0;
170 DIM Xb(300),Yb(300),Zb(300),Fb(300

190 ON ERROR:PROCError1
200 PROCVariables
210 ON ERROR:PROCError2
220 PROCInitialValues
230 PROCPicture:VDU23,1,0;0;0;0;
240 REPEAT
250 PROCFind
260 PROCDisplay
270 IF INKEY-1:PROCStat:UNTIL FALSE
280 IF INKEY-74 OR Read%=Rmax% PROCSta
t:REPEAT UNTIL INKEY-1
290 UNTIL FALSE
300 END
310 :
1000 DEF PROCVariables
1010 CLS
1020 PRINT'"Write a formula, F, in the
 notation of"'"Basic with three or fewer
 variables"'"selected from X, Y, or Z (U
pper case)."
1030 PRINT'"An example could be F = X+Y
, in which"'"case key in X+Y. Use low va
lues of"'"variables (up to about 10), an
d standarddeviations of about 1 to 10 pe
rcent."
1040 PRINT'"Values outside these may be
 accepted"'"but the picture may be poor.
"
1050 INPUT'"Write the wanted formula i
n Basic"'"notation: "Val$:X=1.132:Y=1.45
3:Z=1.576:Test=EVAL(Val$)
1060 PRINT'"Give the mean and standard
deviation"'"of each variable."
1070 IF INSTR(Val$,"X")<>0 INPUT'"What
is the mean of X? "Xmn$:INPUT'"and the s
tandard deviation of X? "Xdv$:ELSE Xmn$=
"":Xdv$=""
1080 IF INSTR(Val$,"Y")<>0 INPUT'"What
is the mean of Y? "Ymn$:INPUT'"and the s
tandard deviation of Y? "Ydv$:ELSE Ymn$=
"":Ydv$=""
1090 IF INSTR(Val$,"Z")<>0 INPUT'"What
is the mean of Z? "Zmn$:INPUT'"and the s
tandard deviation of Z? "Zdv$:ELSE Zmn$=
"":Zdv$=""
1100 INPUT'"How many trials? "Rmax%:IF
Rmax%=0 Rmax%=1E8
1110 XM%=TRUE:YM%=TRUE:ZM%=TRUE
1120 Xmn=VALXmn$:IF Xmn$="" XM%=FALSE
1130 Xdv=VALXdv$:IF Xdv$="" XM%=FALSE
1140 Ymn=VALYmn$:IF Ymn$="" YM%=FALSE
1150 Ydv=VALYdv$:IF Ydv$="" YM%=FALSE
1160 Zmn=VALZmn$:IF Zmn$="" ZM%=FALSE
1170 Zdv=VALZdv$:IF Zdv$="" ZM%=FALSE
1180 ENDPROC
1190 :
```

```
1200 DEF PROCInitialValues
1210 IF INSTR(Val$,"X")<>0 AND XM%=0 OR
INSTR(Val$,"Y")<>0 AND YM%=0 OR INSTR(V
al$,"Z")<>0 AND ZM%=0 PRINT'"Not all of
the means and S.D.s have beengiven. Plea
se run the program again.":END
1220 X=Xmn:Y=Ymn:Z=Zmn:Fmn=EVAL(Val$)
1230 Max=Xmn
1240 IF Ymn>Max Max=Ymn
1250 IF Zmn>Max Max=Zmn
1260 IF Fmn>Max Max=Fmn
1270 Hscl = 800/Max
1280 Xmpos=(Xmn*Hscl+140)
1290 Ympos=(Ymn*Hscl+140)
1300 Zmpos=(Zmn*Hscl+140)
1310 Fmpos=(Fmn*Hscl+140)
1320 Fsum=0:Fsq=0
1330 ENDPROC
1340 :
1350 DEF PROCPicture
1360 CLS
1370 PRINT'"  X,Y,Z and F plotted horiz
ontally, and  their frequencies vertica
lly."
1380 PRINT"  Repeated values of F are c
alculated,"'"  where  F  =  ";Val$
1390 IF XM% PRINT'"  X:  Mean ";Xmn;" a
nd S.D. ";Xdv
1400 IF YM% PRINT"  Y:  Mean ";Ymn;" an
d S.D. ";Ydv
1410 IF ZM% PRINT"  Z:  Mean ";Zmn;" an
d S.D. ";Zdv
1420 PRINT'"  RETURN to halt, or SHIFT
to continue."':H%=POS:V%=VPOS:PRINTTAB(1
[illegible]
1440 MOVE 20,80:DRAW 1259,80
1450 MOVE 140,65:DRAW 140,95
1460 MOVE 125,60:VDU5:PRINT"o"
1470 IF XM% MOVE Xmpos-35,60:PRINT"X";
1480 IF YM% MOVE Ympos-15,60:PRINT"Y";
1490 IF ZM% MOVE Zmpos+5,60:PRINT"Z";
1500 MOVE Fmpos-15,30:PRINT"F":VDU4
1510 W%=1032-(V%+4)*32
1520 VDU24,20;0;1259;W%;
1530 MOVE 20,80:DRAW 20,W%
1540 DRAW 1259,W%:DRAW 1259,80
1550 ENDPROC
1560 :
1570 DEF FNGauss:SS=0:FOR N%=1 TO 12
1580 SS=SS+RND(1):NEXT:=SS-6
1590 :
1600 DEF PROCFind
1610 Read%=Read%+1
1620 IF XM% X=Xmn+Xdv*FNGauss
1630 IF YM% Y=Ymn+Ydv*FNGauss
1640 IF ZM% Z=Zmn+Zdv*FNGauss
1650 FF=EVAL(Val$):Fsum=Fsum+FF
1660 Fsq=Fsq+FF*FF
1670 VDU31,H%+32,V%:@%=0
1680 PRINT;"[";Read%;"]";:@%=&02050A
1690 ENDPROC
1700 :
1710 DEF PROCDisplay
1720 IF XM% Xpos=X*Hscl+140:Xs%=INT((Xp
os-Xmpos)/4)+150:IF Xs%>299 OR Xs%<1 END
PROC
1730 IF YM% Ypos=Y*Hscl+140:Ys%=INT((Yp
os-Ympos)/4)+150:IF Ys%>299 OR Ys%<1 END
PROC
1740 IF ZM% Zpos=Z*Hscl+140:Zs%=INT((Zp
os-Zmpos)/4)+150:IF Zs%>299 OR Zs%<1 END
PROC
1750 Fpos=FF*Hscl+140:Fs%=INT((Fpos-Fmp
os)/4)+150:IF Fs%>299 OR Fs%<1 ENDPROC
1760 IF XM% Xb(Xs%)=Xb(Xs%)+1:Xn=Xb(Xs%
)*Vscl:MOVE Xpos,80:DRAW Xpos,80+Xn
1770 IF YM% Yb(Ys%)=Yb(Ys%)+1:Yn=Yb(Ys%
)*Vscl:MOVE Ypos,80:DRAW Ypos,80+Yn
1780 IF ZM% Zb(Zs%)=Zb(Zs%)+1:Zn=Zb(Zs%
)*Vscl:MOVE Zpos,80:DRAW Zpos,80+Zn
1790 Fb(Fs%)=Fb(Fs%)+1:Fn=Fb(Fs%)*Vscl
1800 MOVE Fpos,80:DRAW Fpos,80+Fn
1810 ENDPROC
1820 :
1830 DEF PROCStat
1840 @%=&02050A
1850 Fmn=Fsum/Read%
1860 Fdv=SQR(Fsq/Read%-Fmn^2)
1870 PRINT'"  F:  Mean ";Fmn;" and S.D
. ";Fdv;"."
1880 @%=0:VDU31,H%+22,V%
1890 PRINT;Read%;''
1900 ENDPROC
1910 :
1920 DEF PROCError1
1930 IF ERL=1050 PRINT'"The Function wa
s not well defined."'"Check that only t
he capital letters"'"X, Y and Z were use
d as variables."'"Press Return to try ag
ain.":IF GET:RUN:ELSE PROCError2
1940 ENDPROC
1950 :
1960 DEF PROCError2
1970 CLS:VDU26:@%=&A:IF ERR=17 END:ELSE
REPORT:PRINT" at line ";ERL:END
1980 ENDPROC
```

Learning about Chaos

Mike Williams reviews a book on the theory of chaos

Chaos: Making a New Science
by James Gleick,
published in Cardinal by Sphere Books, 1989 at £5.99.

We have all become accustomed to the likes of the Mandelbrot set and Julia sets appearing in computer magazines from time to time (and that includes BEEBUG). The fascinating visuals that result have a powerful ability to intrigue and mystify as order appears out of apparent chaos.

This month's main feature program addresses the subject more directly, and provides readers with programs with which to explore a further aspect of the theory of chaos. The book reviewed here complements the program and provides a comprehensive and popular treatise of the whole subject. It is currently being supplied by Logotron as part of an educational software pack on the same subject for the Archimedes.

For a start, this is not an easy book to describe or summarise. That may be the consequence of the American nationality of its author, as the book, in my view contains a substantial amount of redundant material and somewhat boring biographical detail. For example, one learns of Mandelbrot's parents, Lithuanian Jews, earning a living in '20s Warsaw as clothing wholesaler and dentist! There is also an account of how Mandelbrot came to invent the word *fractal* while looking at his son's Latin dictionary!

The book begins by describing some early experiments in theoretical numerical weather forecasting, which led ultimately to the realisation that infinitely small changes can have significant consequences. This is referred to as the *butterfly effect*, "the notion that a butterfly stirring the air today in Peking can transform storm systems next month in New York". Scientists had previously assumed that order prevailed and that all systems were describable ultimately, in mathematical deterministic terms. The theory of chaos shows that even apparently ordered systems can degenerate into chaos, often as a result of minute changes, and yet out of such chaos order can eventually reappear. This is demonstrated by the work of Robert May, whose studies of population growth form the basis of the feature in this issue of BEEBUG referred to earlier.

Another concept on which the book dwells at length is how delving deeper and deeper into detail reveals yet more of the same. The classic example is that of a coastline - no matter how much you magnify it, it continues to exhibit the same level of indentation and roughness - a measurement known as the *fractal dimension*. This is true of the Mandelbrot set where magnifying any small part reveals yet more detail in a never ending quest.

There is more, much more, to be found in this book which is a mine of fascinating information, despite my criticism of its inclusion of too much unwanted, largely biographic, detail. English readers might also find fault with its largely transatlantic outlook and the way the whole treatise seems to be woven around the lives of the human participants.

Despite all of that, I found much that was of interest, providing a welcome background and context to what had previously been fragmented (fractal?) glimpses of the world of chaos described therein. The book runs to some 350 pages with 8 pages of glossy full colour illustrations, plus many black and white drawings. There are also over 300 notes on the sources on which the book is based, including many further references for those wishing to pursue original material.

If you are at all interested in broadening your knowledge of the Mandelbrot set and the world of ordered chaos of which it is a part, then I recommend this book despite my reservations about some aspects of its style, and it's certainly good value for the price.

Printing Characters 128 to 255

set on a model B, and to download it to a printer.

Many printers have printable characters in the ASCII range 128-255. Often this range consists of an italicised version of the normal set (32-126); sometimes it contains the standard IBM character set; and some printers can be switched between the two.

Master series computers have an entirely different set of characters in this range. If you type in the following line you will see them displayed on the screen:

```
FOR A%=128 to 255:P.;A%"  "CHR$A%:N.
```

It can be useful to have a program which will print this character set on a printer, and the listing given here is designed to do just that. It is called WYPIWYS (What You Print Is What You See), and should work with any Epson-compatible printer which has the facility to download character definitions into the printer's memory (normally using the control sequence ESC "&"). Unfortunately an exact match is not possible as the Master set uses an 8x8 matrix for each character, whereas those in the printer are arranged in an 8x11 matrix, but with the requirement that adjacent points in any row cannot be printed.

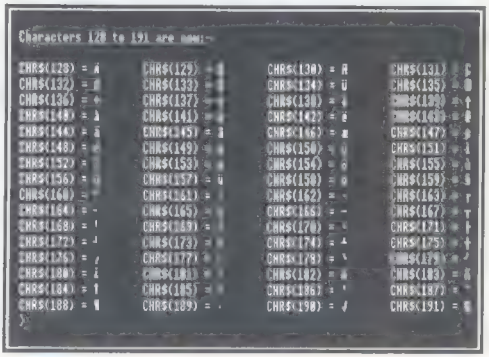
BBC MODEL B

The model B does not have the extended character set of the Master, but in its default state it is possible for characters 128-159 to be user-defined, using the VDU23 statement (see the User Guide). Characters 160-255 can also be defined, by allocating extra RAM in chunks of one page (of 256 bytes) for each block of 32 character definitions. This requires the value of PAGE to be increased, so there has to be a trade-off between memory usage and the number of characters you define.

The WYPIWYS program includes a procedure (PROCdefineB) and a set of DATA statements which re-define *all* the characters from 128 to 255 on a model B to be the same as on a Master. However, you can choose how many characters to re-define to suit your own requirements. Each DATA statement contains 2 character definitions, in the form of an 8-byte hex string, starting at characters 128-129 in line 1610. The fewer characters you define, the fewer lines you have to type in! Simply alter last% in line 140 to the highest character to be defined, and include the appropriate number of DATA statements when you type in the listing. Then *before loading the program* set the value of PAGE as follows:

```
last%=159    no alteration to PAGE
last%=191    PAGE=PAGE+&100
last%=223    PAGE=PAGE+&200
last%=255    PAGE=PAGE+&300
```

When the characters have been defined, PROCmodelBtest displays them on the screen. Master owners need not type in these two procedures, or the DATA statements!



Part of the Master's extended character set reproduced on a model B

One instance in which this program may be useful is when printing out program listings which contain teletext control characters. These are sometimes used in REM statements, or in PRINT statements where insertion of a control character, by using Shift or Ctrl with a function key, is easier than typing in, say, 'CHR$129'. Normally when such listings are printed the results can be chaotic, with form feeds, carriage returns and other control codes being sent to the printer.

PROGRAM OUTLINE

PROCsetup first sends the appropriate codes to the printer to notify it that characters are to be downloaded. It then repeatedly calls PROCreadchar for each character, which reads the character definition using OSWORD 10 and translates it from the horizontal bit-pattern supplied into the vertical bit-pattern required by the printer. The character is then downloaded to the printer before calling PROCreadchar for the next one.

When all the characters have been downloaded, PROCtest causes the printer to print out all the characters from 128 to 255 twice; firstly the original character from the printer's own set, and secondly the newly-defined character.

PROCdefineB first makes a *FX20 call, with X set to 0,1,2 or 3 depending on the number of characters to be defined. The procedure then simply unpacks the hex strings in the DATA statements, and uses VDU23 to re-define the characters.

```
   10 REM Program WYPIWYS
   20 REM Version B1.0
   30 REM Author  Lance Vick
   40 REM BEEBUG  March 1990
   50 REM Program Subject to copyright
   60 :
  100 MODE3
  110 ON ERROR:VDU3:WIDTH0:REPORT:PRINT"
 at line ";ERL:END
  120 master=(INKEY(-256)=253)
  130 first%=128:last%=255
  140 IF NOT master:last%=255
  150 IF NOT master:PROCdefineB
  160 IF NOT FNonline:PROCswitchon
  170 PRINT'"WYPIWYS or"''"What You Prin
t Is What You See"'
  180 PROCsetup:PROCtest
  190 END
  200 :
 1000 DEF PROCsetup
 1010 PRINT'"Redefining Characters 128 t
o ";last%
 1020 VDU2,1,27,1,&3A,1,0,1,0,1,0
 1030 VDU1,27,1,&36
 1040 VDU1,27,1,&26,1,0,1,first%,1,last%
 1050 D%=&70:B%=&79
 1060 X%=D%MOD256:Y%=D%DIV256:A%=10
 1070 FOR C%=first% TO last%
 1080 VDU1,&8B
 1090 PROCreadchar(C%)
 1100 VDU1,(B%?0 ORB%?1),1,0,1,B%?2,1,0,
1,(B%?3 OR B%?4),1,0,1,B%?5,1,0,1,B%?6,1
,0,1,B%?7
 1110 NEXT:VDU3:ENDPROC
 1120 :
 1130 DEF PROCreadchar(I%)
 1140 ?D%=I%:CALL &FFF1
 1150 FOR N%=7 TO 0 STEP-1
 1160 V%=0:P%=2^N%
 1170 FOR M%=0 TO 7
 1180 V%=V%-2^M%*((D%?(8-M%) AND P%)>0)
 1190 NEXT:B%?(7-N%)=V%
 1200 NEXT:ENDPROC
 1210 :
 1220 DEF PROCtest
 1230 VDU2:WIDTH80
 1240 PRINT'"Characters 128 to ";last%
 1250 PRINT"First Character is from Prin
ter's ROM"'
 1260 PRINT"Second Character is Redefine
d version"'
 1270 FOR C%=first% TO last%
 1280 PRINT"CHR$(";C%") = ";
 1290 FOR I%=0 TO 1
 1300 VDU1,27,1,&25,1,I%,1,0
 1310 PRINTCHR$(C%)SPC3;
 1320 NEXT:NEXT
 1330 PRINT:VDU3:WIDTH0
 1340 ENDPROC
 1350 :
 1360 DEF PROCdefineB
 1370 RESTORE
 1380 FOR C%=first% TO last%:VDU23,C%
 1390 READ A$:FOR I%=0 TO 7
 1400 VDU EVAL("&"+MID$(A$,I%*2+1,2))
 1410 NEXT:NEXT
 1420 PROCmodelBtest
 1430 ENDPROC
 1440 :
 1450 DEF PROCmodelBtest
 1460 PRINT'"Characters 128 to "STR$last
%" are now:-"'
 1470 FOR C%=first% TO last%
 1480 PRINT"CHR$(";C%") = "CHR$(C%)SPC7;
 1490 NEXT:ENDPROC
 1500 :
 1510 DEF FNonline:*FX21,3
 1520 VDU2,1,0,1,0,3
 1530 =(ADVAL-4=63)
 1540 :
 1550 DEF PROCswitchon
 1560 REPEAT:PRINT'"Printer not on-line"
 1570 PRINT'"Switch on and press Space"
 1580 REPEAT UNTIL GET=32
 1590 UNTIL FNonline:ENDPROC
 1600 :
 1610 DATA 66003C667E666600,3C663C667E666600
 1620 DATA 3F66667F66666700,3C666060606 3C60
 1630 DATA 0C187E607C607E00,663C666666663C00
```

Improving Basic Programs (Part 2)

by Mike Williams

Last month I outlined some of the ways in which you can improve your programming in Basic, and I dealt at some length with two examples. The first was concerned with enhancing the interface between the program and the user - making the screen display more slick and professional where input is concerned. The second example was concerned with redundancy, and I want to consider that further this month.

By redundancy I mean locating areas within a program which can be written more concisely than might at first seem possible. The example which I discussed last time showed how a sequence of IF statements can often be reduced to a single such statement, provided that the conditions involve testing the same variable each time. Of course, condensing a program in this way may result in something which is less readable, where the original intention (i.e. the set of individual tests) has become obscured, but the concept is well worth remembering, and if a program is tight on memory space well nigh essential.

When compiling the Postbag page for this issue, I was reminded of another technique which is far from obvious. The letter, if you want to refer to it, shows a method for defining a 'plus-or-minus' sign, as well as correct pound (£) and hash (#) signs for a printer. In this case, a FOR-NEXT loop easily copes with the requirements, but what if you have a similar repetitive pattern, but one which is not quite so regular?

Suppose, as an example, we want to output to a printer in graphics mode, rather than in plain text. To do this, it is necessary to send codes to the printer which say we are printing graphics (Escape-K, where K has ASCII code 75), how many bytes of information are to follow (as two bytes n1 and n2), and then the bytes themselves, for example:

```
VDU1,27,1,75,1,11,1,0,1,135,1,140,1,100,
1,54,1,30,1,40,1,0,1,54,1,0,1,40,1,0,1,13,1,10
VDU1,27,1,75,1,10,1,0,1,196,1,110,1,112,
1,16,1,14,1,22,1,30,1,16,1,130,1,68,1,13,1,10
VDU1,27,1,75,1,10,1,0,1,182,1,120,1,100,
1,34,1,34,1,34,1,250,10,34,1,34,1,34,1,13,1,10
```

Don't worry about the precise effect of this - it is unimportant as far as this article is concerned. Note the alternate '1' in all three of the above statements. In a VDU instruction, this ensures that the following value is sent *only* to the printer, not to the screen as well. Printing graphics (and on other occasions) can involve values which, if output to the screen, would have undesirable effects.

Be that as it may, it seems a bit of a waste to specify each '1' individually. Why not, I thought, have a loop which we go round three times, once for each VDU statement, inside which we have another loop which reads the relevant codes from a DATA statement, and sends them to the printer. Thus a simple instruction like:

```
READ C%
VDU1,C%
```

could do all the work.

As a further refinement, we can see that each VDU statement begins with the sequence '1,27,1,75' - which sends 'Escape-K' to the printer - so why bother putting that in with the other data; why not put it in the program? We can treat similarly the 1,13,1,10 (carriage return, line feed) which occurs at the end of each line - to move the printer to the start of the next line. In principle, our revised routine will look like this:

```
FOR I%=1 TO 3:VDU1,27,1,75
FOR J%=1 TO 13
READ C%:VDU1,C%
NEXT J%:VDU13,10:NEXT I%
DATA 11,0,135,140,100,54,30,40,0,54,0,40,0
DATA 10,0,196,110,112,16,14,22,30,16,130,68
DATA 10,0,182,120,100,34,34,34,250,34,34,34
```

Of course, you may argue (and perhaps rightly) that the original version was much simpler, and easier to understand. I don't disagree, in this case, but it is the idea which is important.

However, if you look at the original VDU statements, you will see that the first contains 13 values to be sent to the printer (after the initial codes 27,75), while the other two each have 12 values. So our inner loop, in which the variable J% counts from 1 to 13 will be wrong for the second two VDU statements.

That might suggest that all our clever ideas will come to naught. However, that is not so, and although we have taken a little while to reach this point, it is the crux of what I am talking about. Suppose we want the FOR-NEXT loop which uses J% to count from 1 to 11 the first time and from 1 to 10 each of the other two times. How can this be accomplished?

The solution depends upon the fact that on any BBC computer the values of TRUE and FALSE are represented by the values -1 and 0 respectively. Thus although we may think of a condition like 'I%>1' as being either true or false, it also has a value of either -1 or 0, and that value can be used as such. For example we could write:

```
13+(I%>1)
```

This would have the value 13 (if I% is not greater than 1, so that 'I%>1' is false with a value 0), or 12 (if 'I%>1' is true with a value of -1).

This is just what we need in the example described above, so we must replace the second line of the routine with:

```
FOR J%=1 TO 13+(I%>1)
```

This idea of using the *value* of a condition as part of an expression is one that is well worth knowing, so we'll look at another example before we finish.

Suppose you are writing a graphics program which is to display a small cross on the screen (representing the current position), which is to be moved using the cursor keys. We will make the cross 16 graphics units high and 16 graphics units wide (remember, the graphics screen is 1280 units horizontally (from 0 to 1279), and 1024 vertically (from 0 to 1023).

To make the cross appear to move, according to the pressing of the cursor keys, we will need to use Exclusive OR plotting, drawing the cross once to make it appear, drawing it a second time to make it disappear, and then redrawing it in a new position. We will also assume that we start from the centre of the screen. In outline, the routine will look something like this:

```
*FX4,1
GCOL 3,1
X%=640:Y%=512
REPEAT
PROCcross(X%,Y%)
PROCmove
PROCcross(X%,Y%)
UNTIL FALSE
```

The *FX call disables cursor editing and causes the cursor keys to generate ASCII codes (136 for *left*, 137 for *right*, 138 for *down* and 139 for *up*). The GCOL statement selects the colour for plotting (this is the second value - the result will depend on the mode used, but see later), while the first value of 3 selects Exclusive OR plotting (I won't explain the precise details of this; just accept that it does what I say). The program then sets X% and Y% to the agreed initial values.

We then have a REPEAT-UNTIL loop involving three procedure calls (the procedures have yet to be written). PROCcross draws a cross on the screen. The first time it is called the cross will appear, the second time it will disappear, and so on. The procedure PROCmove checks to see if a cursor key has been pressed and updates X% and Y% accordingly.

Incidentally, this approach to programming is a well recognised technique in its own right. Get the broad outline of the program written first, using procedure or function calls where necessary to gloss over the smaller detail you don't want to bother about yet. When you are satisfied that all is well, start looking at those procedures, writing the definitions in more detail.

Now back to our own program, and before we proceed to fill out the two procedures, there is an anomaly that must be taken care of. The second call to PROCcross is intended to remove the cross from the screen by re-drawing it in its original position. But PROCmove may have already updated the values of X% and Y% to reflect a new position. We need to introduce two additional variables, which I will call *oldX%* and *oldY%* to remember the previous

values of X% and Y%. The main loop will; now look as follows:

```
REPEAT
PROCcross(X%,Y%):oldX%=X%:oldY%=Y%
PROCmove
PROCcross(oldX%,oldY%)
UNTIL FALSE
```

Now we can proceed. First of all PROCcross can be written as follows:

```
DEF PROCcross(X%,Y%)
MOVE X%-8,Y%:DRAW X%+8:Y%
MOVE X%,Y%+8:DRAW X%,Y%-8
ENDPROC
```

This really needs no further explanation. Now for the second procedure, and this is where the idea that I described previously will come into play. First of all we need to decide something else.

Although the graphics screen runs from 0 to 1279 horizontally and 0 to 1023 vertically, we cannot literally move one graphics unit at a time. There is always a minimum step size which depends on the mode in use. If you specify a smaller step size than is possible on the visible screen, you just stay where you are. The vertical step size is always 4. The horizontal step size can be 2 (mode 0), 4 (modes 1 and 4) or 8 (modes 2 and 5) - the other modes are not graphics modes anyway. We will assume we are using either mode 1 or mode 4, so that the minimum step size is 4 in both directions. If you are using a different mode then adjust the horizontal step size in what follows.

The most obvious way of coding PROCmove is as follows:

```
DEF PROCmove
S%=GET
IF S%=136 THEN X%-X%-4
IF S%=137 THEN X%-X%+4
IF S%=138 THEN Y%=Y%-4
IF S%=139 THEN Y%=Y%+4
IF X%<0 THEN X%=0
IF X%>1276 THEN X%=1276
IF Y%<0 THEN Y%=0
IF Y%>1020 THEN Y%=1020
ENDPROC
```

Perhaps you were initially surprised by the last four IF statements, but we have to make sure that the centre of our cross remains on the screen (you might, of course, want to have a more restricted screen area). The reason for 1276 (not 1279) derives from the fact that, moving in steps of 4, the highest value we can reach that is on the screen is 1276. Similarly, the highest on-screen value of the y screen co-ordinate is 1020.

Now let's see how the first two statements might be rewritten using the numerical value of the test condition:

```
X%=X%+4*(S%=136)-4*(S%=137)
```

and similarly for Y%:

```
Y%=Y%+4*(S%=138)-4*(S%=139)
```

Remember that only one value for S% is possible at any one time. If you are not sure about these two lines, try each possible value of S% in turn, substituting 0 or -1 as appropriate for each condition. For example, if S%=136 (cursor left), then the first line becomes:

```
      X%=X%+4*(-1)-4*(0)
i.e.  X%=X%-4-0
thus: X%=X%-4
```

So far so good, and if some other key should be pressed on the keyboard, then our revised (and original) version will ensure that the values of X% and Y% remain unchanged.

However, we can take the idea of using logical values even further. No matter how long a logical condition becomes, the resulting value is always either -1 or 0. So we can incorporate the last four IF statements into our revised first two lines as follows:

```
X%=X%+4*(S%=136 AND X%>0)-4*(S%=137 AND
X%<1276)
Y%=Y%+4*(S%=138 AND Y%>0)-4*(S%=139 AND
Y%<1020)
```

When AND is used to join two conditions together, the result is true only if both of the individual conditions are true (-1); in all other cases the final result is false (0). Again, I suggest you try evaluating either expression above in particular cases to verify their validity. Moreover, we have reduced our original eight lines to a mere two (albeit longer), not bad eh?

Using conditions as values can take a bit of getting used to, but it can streamline your programs as I have shown, and is the kind of 'trick' that can make you feel quite satisfied when you can put it to good use, even if the new version is sometimes less readable.

EdiKit (Part 3)

This line continues the development of a programmer's toolkit aimed for use with Basic programs. EDIKIT3 adds five more commands to the EDIKIT ROM.

ENTERING THIS MONTH'S PROGRAM

There are some differences between the Basic II (on the model B) and Basic IV (Master) versions of this program. Note the address of insline in lines 1030 and 1040; &BC8D for Basic II and &BAEB for Basic IV. Line 4200 should be omitted if you are going to run the program with Basic IV; the whole of the routine tokline (lines 4590 to 4660) is also unnecessary with Basic IV. Where appropriate, substitute your own SWR commands in lines 170 and 180.

Save this month's program as EDIKIT3. Before running it, ensure that the ROM image from parts 1 and 2(Edirom &8000 to &9240) is already loaded in a vacant sideways RAM slot, and that in lines 170 and 180 you replace 'X' by the same slot number (as last month). When you run EDIKIT3 answer Y to the prompt "SAVE and LOAD ROM?"; Edirom3 will be overlaid in slot 'X' from &9200 to &9630 and the accumulated ROM from &8000 to &9830 will be re-saved as Edirom. Now press Ctrl-Break to initialise the ROM. *H. EDIKIT should print a list of commands and their formats.

USING THE NEW COMMANDS

***LFROM <element>** lists eight lines of program beginning with the first occurrence of the specified element. An element of Basic is a name (of a variable, function or procedure), a Basic keyword, a numeric value, a quote string, an OS star command, or an assembler label, opcode or operand. All the rules about wildcards and abbreviations which apply to *FBASIC also apply to *LFROM; for further details see the notes on EDIKIT2 in the last issue of BEEBUG Magazine (Vol.8 No.8). At the end of the eight lines, you may choose either to continue listing more of the program (press Shift), or to start again at the next occurrence of the specified element (press Tab). You may quit by pressing Escape. This command can be very useful for listing FOR loops, for example; just enter *LF.F. (note the abbreviations of *LFROM FOR).

***LPROC <procname>** (which may be shortened to *LP.<procname>) lists the named procedure starting at the line in which the DEFPROC occurs and ending at the first ENDPROC.

***LFN <fnname>** (no abbreviation) lists out the function with the specified name.

***RTEXT /<string1>/<string2>/** and ***RBASIC <element1>/<element2>** are find and replace commands directly analogous to FTEXT and FBASIC. The commands may be abbreviated to *RT. and *RB. Both look for occurrences of the first parameter, and print out in turn each line which contains it. A pointer indicates the last letter of the target word so that multiple occurrences in one line will not be confused.

You are offered the option of replacing the first parameter with the second (press R), skipping over the current instance (press S), or aborting the run by pressing Escape. If you choose to make the substitution, the amended line is printed again enabling you to check that the desired effect has been achieved. Note that the slash character is the normal delimiter for the two parameters of RTEXT, but any other character may be used as a delimiter (as for FTEXT) so long as that character occurs three times; otherwise an error will be signalled. The use of three delimiters means that initial or final spaces may be included in either parameter. Note that the parameters to RBASIC are also separated by a slash character but this may not be varied; any leading or trailing spaces are omitted.

Parameters to RBASIC may consist of any element of Basic as defined in the discussion of EDIKIT2. Keyword abbreviations may be used but the star wildcard is not allowed; an error message will be printed if you try. To substitute one line number for another (following GOTO etc.) you need only use the ^ character as a marker in the case of the first parameter. Thus *RB.^2000/3300 will replace the line number 2000 with 3300.

Attempts to call any of the remaining commands (VARLIST, SYSVARS, FKDEFS, PCOMM ... ZCOMM) will result in a "Not implemented" message at this stage.

```
  10 REM Program EDIKIT3
  20 REM Version 1.01
  30 REM Author  Bill Hine
  40 REM BEEBUG  March 1990
  50 REM Program subject to copyright
  60 :
  100 MODE 7
  110 start=&9200:size=&600:DIMcode &630
  120 PROCinitvars:PROCassemble
  130 PROClinks
  140 PRINT"SAVE and LOAD ROM? Y/N"
  150 A=GET AND&DF:IFA<>ASC"Y"THENEND
  160 OSCLI("SAVE edirom3 "+STR$~code+"+
"+STR$~(O%-code))
  170 *SRLOAD edirom3 9200 X
  180 *SRSAVE edirom 8000 9830 X
  190 PRINT"Press CTL BRK to initialise"
  200 END
  210 :
 1000 DEF PROCinitvars
 1010 ipbuff=&600:ipbuff2=&680
 1020 buffer=&700
 1030 insline=&BC8D:REM BASIC II
 1040 REM    =&BAEB:REM BASIC IV
 1050 page=&1D:top=&12:commandln=&F2
 1060 nl=&0D:cr=nl:rem=&F4:def=&DD
 1070 proc=&F2:endproc=&E1:fn=&A4:if=&E7
 1080 else=&8B:quote=ASC"""":spc=ASC" "
 1090 esck=&8F:tabk=&9F:shiftk=&FF
 1100 :
 1110 oswrch=&FFEE:osnewl=&FFE7
 1120 osbyte=&FFF4:osword=&FFF1
 1130 oswrch=&FFE3:osrdch=&FFE0
 1140 romexit=&8803:escexit=&8806
 1150 initscanft=&8809:scanprog=&880C
 1160 detokline=&880F:prntline=&8812
 1170 prnttext=&8815:notimp=&8818
 1180 initbptr=&881B:initpptr=&881E
 1190 reinitbptr=&8824:prnteoprog=&882A
 1200 initscanfb=&8C3F:initscanfkw=&8C42
 1210 ipbasparam=&8C45:testdelim=&8C48
 1220 testopcode=&8C4B
 1230 :
 1240 flags=&70:ay=&71:ex=&72:wy=&73
 1250 strlen=&74:str2len=&75:token=&76
 1260 tokn=&77:hitokn=&78:starttokn=&79
 1270 endtokn=&7A:bptr=&80:pptr=&84
 1280 scantype=&88:action=&8A
 1290 ipdelim=&8C:diff=&8D:width=&8E
 1300 scrollmode=&8F
 1310 ENDPROC
 1320 :
 1330 DEF PROCassemble
 1340 :
 1350 FOR pass=4 TO 7 STEP3
 1360 P%=start:O%=code:[OPT pass
 1370 :
 1380 .proccalls
 1390 JMP lfrom:JMP lproc:JMP lfn
 1400 JMP rbasic:JMPrtext
 1410 .sysinf JMP &9800:.varlist JMP &98
:
 1420 .fkdefs JMP &9806:.pcomm JMP &9809
 1430 .qcomm JMP &980C:.rcomm JMP &980F
 1440 .scomm JMP &9812:.tcomm JMP &9815
 1450 .ucomm JMP &9818:.vcomm JMP &981B
 1460 .wcomm JMP &981E:.xcomm JMP &9821
 1470 .ycomm JMP &9824:.zcomm JMP &9827
 1480 :
 1490 .lfrom
 1500 JSR svscrollmode:LDA#0:STA flags
 1510 JSR ipbasparam:JSR initbptr
 1520 LDA#lfrnextmatch MOD256:STAaction
 1530 LDA#lfrnextmatch DIV256
 1540 STAaction+1:JSR initscanfb
 1550 JSR testopcode
 1560 JSR scanprog:JSR prnteoprog
 1570 LDA scrollmode:JSR oswrch
 1580 LDA#15:LDX#1:JSR osbyte:JMPromexit
 1590 :
 1600 .lfrnextmatch
 1610 LDAbptr:PHA:LDAbptr+1:PHA:TYA:PHA
 1620 JSR prnttext
 1630 EQUS"ESC:quit":EQUB spc:EQUB spc
 1640 EQUS"TAB:next match":EQUB spc
 1650 EQUB spc:EQUS"SHIFT:more":EQUW nl
 1660 LDX#8:JSR lfrlistlp
 1670 PLA:TAY:PLA:STAbptr+1:PLA:STA bptr
 1680 JSR detokline:RTS
 1690 :
 1700 .lfrlistlp
 1710 TXA:PHA:JSR ptbasline:PLA:TAX
 1720 JSR reinitbptr:LDY#1:LDA(bptr),Y
 1730 CMP#&FF:BEQ lfroptns2
 1740 DEX:BNE lfrlistlp:JSR lfroptns
 1750 BEQlfrexit:LDX#8:JMP lfrlistlp
 1760 .lfrexit RTS
 1770 :
 1780 .lfroptns .ktloop
 1790 LDX#esck:JSR keytest:CPX#&FF
 1800 BEQ escpress:LDX#tabk:JSR keytest
 1810 CPX#&FF:BEQ tabpress:LDX#shiftk
 1820 JSR keytest:CPX#&FF:BNE ktloop
 1830 .shiftpress LDA#1:RTS
 1840 .tabpress LDA#0:RTS
 1850 .escpress JMP escexit
 1860 .lfroptns2
 1870 JSR prnttext
```

```
1880 EQUS"End of prog - ESC:quit "
1890 EQUS" TAB:next match":EQUWnl
1900 .ktlp2
1910 LDX#esck:JSR keytest:CPX#&FF
1920 BEQ escpress:LDX#tabk:JSR keytest
1930 CPX#&FF:BEQ tabpress:JMP ktlp2
1940 .keytest
1950 TXA:PHA:LDA#&81:LDX#2:LDY#0
1960 JSR osbyte:PLA:TAX
1970 LDA#&81:LDY#&FF:JMP osbyte
1980 .svscrollmode
1990 LDA#&75:JSR osbyte
2000 TXA:AND#4:LSRA:LSRA:STA ex
2010 LDA#15:JSR oswrch
2020 SEC:SBC ex:STA scrollmode
2030 RTS
2040 :
2050 .lproc
2060 LDA#proc:STA starttokn
2070 LDA#endproc:STA endtokn
2080 JSR lprocfn:JMP romexit
2090 :
2100 .lfn
2110 LDA#fn:STA starttokn
2120 LDA#ASC"=":STA endtokn
2130 JSR lprocfn:JMP romexit
2140 :
2150 .lprocfn LDA#0:STAflags
2160 JSRipbasparam:JSRinitbptr:LDA#def
2170 STAtokn:STAhitokn:JSR initscanfkw
2180 LDA#checkpf MOD256:STA action
2190 LDA#checkpf DIV256:STA action+1
2200 JSR scanprog
2210 .lpfexit
2220 RTS
2230 .checkpf
2240 LDA(bptr),Y
2250 CMP starttokn:BEQ checkname
2260 CMP#spc:BNE lpfexit
2270 INY:JMP checkpf
2280 .checkname
2290 INY:LDX#0:LDA(bptr),Y
2300 CMP ipbuff,X:BNE lpfexit
2310 .chkname2
2320 INX:INY
2330 LDA ipbuff,X:CMP#nl:BEQ chkname3
2340 CMP(bptr),Y:BEQ chkname2
2350 JMP lpfexit
2360 .chkname3
2370 LDA(bptr),Y:DEY
2380 JSR testdelim:BNE lpfexit
2390 .namesmatch
2400 INY:LDA(bptr),Y
2410 CMP#spc:BEQ namesmatch
2420 CMP#ASC"(":BEQ brackets
2430 CMP#nl:BEQ nextline
2440 CMP endtokn:BEQ nmexit
2450 CMP#ASC":":BEQ newstatemt
2460 JMP newstatemt+1
2470 .nmexit RTS
2480 .newstatemt
2490 INY:LDA(bptr),Y
2500 CMP#spc:BEQ newstatemt
2510 CMP endtokn:BNE restofstmt
2520 JMP ptbasline
2530 .restofstmt
2540 CMP#nl:BEQ nextline
2550 CMP#ASC":":BEQ newstatemt
2560 CMP#if:BEQ ifloop
2570 INY:LDA(bptr),Y:JMP restofstmt
2580 .brackets
2590 LDX#0
2600 .braklp
2610 INY:LDA(bptr),Y
2620 CMP#nl:BEQ nextline
2630 CMP#ASC")":BEQ newstatemt
2640 CMP#ASC"(":BNE braklp:INX
2650 .braklp2
2660 INY:LDA(bptr),Y
2670 CMP#nl:BEQ nextline
2680 CMP#ASC"(":BEQ braklp2-1
2690 CMP#ASC")":BNE braklp2
2700 TXA:DEX:BNE braklp2
2710 JMP braklp
2720 .nextline
2730 JSR ptbasline:INY:LDA(bptr),Y
2740 CMP#&FF:BEQ eoprog:JSRreinitbptr
2750 BIT&FF:BMI lpfesc
2760 LDY#3:JMP newstatemt
2770 .eoprog RTS
2780 .lpfesc JMP escexit
2790 .ifloop
2800 INY:LDA(bptr),Y
2810 CMP#nl:BEQ nextline
2820 CMP#ASC":":BEQ newstatemt
2830 CMP#else:BEQ newstatemt
2840 JMP ifloop
2850 :
2860 .rbasic
2870 LDA#0:STA flags
2880 JSR iprbparams:JSR chekwild
2890 JSR initbptr
2900 LDA#options MOD256:STA action
2910 LDA#options DIV256:STA action+1
2920 JSR initscanfb:JSR testopcode
2930 JSR scanprog:JSR prnteoprog
2940 JMP romexit
2950 :
```

```
2960 .iprbparams
2970 LDA(commandln),Y:LDX#0:CMP#ASC"^"
2980 BEQ initlnnoflag:CMP#spc
2990 BNEiprbploop:INY:JMP iprbparams
3000 .initlnnoflag
3010 LDAflags:ORA#4:STAflags:INY
3020 JMP iprbparams
3030 .iprbploop
3040 LDA(commandln),Y:STA ipbuff,X
3050 CMP#nl:BEQ rberr:CMP#ASC"/"
3060 BEQ trspcloop1:INY:INX:CPX#128
3070 BEQtoolong:JMP iprbploop
3080 .trspcloop1
3090 DEX:LDA ipbuff,X:CMP#spc
3100 BNE iprbp2:JMP trspcloop1
3110 .iprbp2
3120 INX:LDA#cr:STA ipbuff,X:TXA
3130 BEQ rberr:STA strlen:INY
3140 .rbploop2
3150 LDA(commandln),Y:LDX#0:INY
3160 CMP#ASC"^":BEQ rbploop2:CMP#spc
3170 BEQ rbploop2:DEY:JMP iprbp2loop
3180 .iprbp2loop
3190 LDA(commandln),Y:STA ipbuff2,X
3200 CMP#nl:BEQ trspcloop2:INY:INX
3210 CPX#128:BEQ toolong:JMP iprbp2loop
3220 .trspcloop2
3230 DEX:LDA ipbuff2,X:CMP#spc
3240 BNE iprbpexit:JMP trspcloop2
3250 .iprbpexit INX:LDA#cr:STAipbuff2,X
3260 STXstr2len:RTS
3270 .rberr
3280 LDX#0:LDA#104:JMP errexit
3290 .toolong
3300 LDA#106:LDX#tltext-detext
3310 JMP errexit
3320 :
3330 .chekwild
3340 JSR initpptr:JSR scanwild
3350 JSR initpptr2:JSR scanwild
3360 .swexit RTS
3370 .scanwild
3380 LDY#0:LDA(pptr),Y:CMP#quote
3390 BEQ swexit:CMP#nl:BEQ swexit
3400 .scanwlp
3410 INY:LDA(pptr),Y:CMP#nl
3420 BEQ swexit:CMP#ASC"*"
3430 BEQ swerr:JMP scanwlp
3440 .swerr
3450 LDA#105:LDX#wetext-detext
3460 JMP errexit
3470 :
3480 .rtext
3490 JSR iprtparams
3500 JSR prnttext:EQUS"Replace ":EQUB0
3510 LDAipdelim:PHA:JSR oswrch
3520 JSR initpptr:LDY#0:JSR ptrtextlp
3530 PLA:PHA:JSR oswrch:JSR osnewl
3540 LDX#3:JSR spaces:JSR prnttext
3550 EQUS"with ":EQUB0:PLA:PHA
3560 JSR oswrch:JSR initpptr2:LDY#0
3570 JSR ptrtextlp:PLA:JSR oswrch
3580 JSRosnewl:JSRosnewl:JSRreplacetxt
3590 JMP romexit
3600 .ptrtextlp LDA(pptr),Y:CMP#nl
3610 BEQ ptrtexit:JSR oswrch:INY
3620 JMP ptrtextlp:.ptrtexit RTS
3630 :
3640 .replacetxt
3650 JSR initbptr:JSR initscanft
3660 LDA#options MOD256:STA action
3670 LDA#options DIV256:STA action+1
3680 JSR scanprog:JMP prnteoprog
3690 :
3700 .iprtparams
3710 LDA(commandln),Y:INY
3720 CMP#spc:BEQ iprtparams
3730 CMP#nl:BEQ rtperr
3740 STA ipdelim:DEY:LDX#&FF
3750 .rtparamslp
3760 INY:INX:CPX#128:BEQ twolong
3770 LDA(commandln),Y
3780 CMP#nl:BEQ rtperr
3790 CMP ipdelim:BEQ p1fin
3800 STA ipbuff,X:JMP rtparamslp
3810 .p1fin
3820 LDA#&0D:STA ipbuff,X:STX strlen
3830 LDX#&FF
3840 .rtparamlp2
3850 INY:INX:CPX#128:BEQ twolong
3860 LDA(commandln),Y:CMP#nl:BEQ rtperr
3870 CMP ipdelim:BEQ p2fin
3880 STA ipbuff2,X:JMP rtparamlp2
3890 .p2fin
3900 LDA#cr:STA ipbuff2,X
3910 STX str2len:RTS
3920 .twolong
3930 JMP toolong
3940 .rtperr
3950 JMP rberr
3960 :
3970 .options
3980 TYA:PHA:JSR prntline:JSRpointer
3990 LDX#5:JSR spaces:JSR prnttext
4000 EQUS"(R)eplace  (S)kip  ESC:quit"
4010 EQUW spc
4020 JSR osrdch:BCS optesc:AND#&DF
4030 JSR oswrch:PHA:JSR osnewl:PLA
```

```
4040 CMP#ASC"R":BEQ doreplace
4050 CMP#ASC"S":BEQ optexit:LDA#7
4060 JSR oswrch:PLA:TAY:JMP options
4070 .optexit
4080 JSR osnewl:PLA:TAY:RTS
4090 .optesc
4100 JMP escexit
4110 .doreplace
4120 LDA str2len:SEC:SBC strlen
4130 JSR movertn
4140 LDAex:SEC:SBC strlen:TAY:LDX#0
4150 .doreplp
4160 CPX str2len:BEQ drexit
4170 LDA ipbuff2,X:STA buffer,Y
4180 INY:INX:JMP doreplp
4190 .drexit
4200 JSR tokline ;  BASIC II only
4210 JSR insertline:JSR detokline
4220 JSR prntline:LDA tokn:BNE optexit
4230 PLA:CLC:ADC diff:TAY:JSR osnewl
4240 RTS
4250 :
4260 .pointer
4270 LDA&30A:SEC:SBC&308:STAwidth
4280 INC width:LDXex:DEX:TXA
4290 .ptrmodlp
4300 CMPwidth:BCC prntptr
4310 SEC:SBCwidth:JMP ptrmodlp
4320 .prntptr
4330 TAX:JSR spaces
4340 LDA#ASC"^":JSRoswrch:JMP osnewl
4350 :
4360 .movertn
4370 STA diff:BEQ mvrtnexit
4380 BPL up:JMP down
4390 .mvrtnexit RTS
4400 .down
4410 LDAex:TAX:CLC:ADC diff:TAY
4420 .movedn
4430 LDA buffer,X:STA buffer,Y
4440 CMP#nl:BEQ movednexit
4450 INX:INY:JMP movedn
4460 .movednexit RTS
4470 .up LDXex
4480 .movetoeoline
4490 LDA buffer,X:CMP#nl:BEQ destindex
4500 INX:JMP movetoeoline
4510 .destindex
4520 TXA:CLC:ADC diff:TAY
4530 .moveup
4540 LDA buffer,X:STA buffer,Y:CPX ex
4550 BEQ moveupexit:DEX:DEY:JMPmoveup
4560 .moveupexit RTS
4570 :
4580 \ Lines 4660-4730 BASIC II only
```

```
4590 .tokline
4600 LDA#buffer DIV256:STA&38:LDA#5
4610 STA&37:SEI:LDA#&51:STA &DE7
4620 LDA#&89:STA&DE8
4630 LDX#0:LDY#&FF:LDA#&BB
4640 JSR osbyte:STX&DE9
4650 LDA#&48:STA&230:LDA#&FF:STA&231
4660 CLI:JMP(&230)
4670 :
4680 .insertline
4690 SEI:LDA#insline MOD256:STA&DE7
4700 LDA#insline DIV256:STA&DE8
4710 LDX#0:LDY#&FF:LDA#&BB
4720 JSR osbyte:STX&DE9
4730 LDY#2:LDA(bptr),Y:STA&2A
4740 DEY:LDA(bptr),Y:STA&2B
4750 LDA#0:STA&2C:STA&2D
4760 LDA#&48:STA&230:LDA#&FF:STA&231
4770 LDY#5:CLI:JMP(&230)
4780 :
4790 .initpptr2
4800 LDA#ipbuff2 MOD256:STA pptr
4810 LDA#ipbuff2 DIV256:STA pptr+1:RTS
4820 :
4830 .ptbasline
4840 JSR detokline:JMP prntline
4850 :
4860 .spaces
4870 INX:LDA#spc
4880 .spcloop
4890 DEX:BEQ spcexit:JSR oswrch
4900 JMP spcloop
4910 .spcexit RTS
4920 .errexit
4930 STA &101:LDA#0:STA &100:LDY#0
4940 .errloop
4950 LDA detext,X:STA &102,Y:INX
4960 INY:CMP#0:BNE errloop:JMP &100
4970 .detext
4980 EQUS"Delimiters?":EQUB 0
4990 .tltext
5000 EQUS"Missing delimiters/Text too l
ong":EQUB 0
5010 .wetext
5020 EQUS"* wildcard not allowed"
5030 EQUB 0
5040 :
5050 ]NEXT:ENDPROC
5060 :
5070 DEF PROClinks
5080 P%=start+size:O%=code+size
5090 FOR I%=0 TO 13
5100 [OPT 7:JMP notimp:]
5110 NEXT:ENDPROC
```

512 Forum

by Robin Burton

After warnings of doom and gloom for some in the last Forum I thought we should look at items of a (slightly) lighter nature this month.

PROGRESS REPORT

I'm not sure that the 512 Technical Guide would normally have a section of its own in the Forum, but I've given it one because it's being mentioned more and more frequently in your letters lately. In case any newer Forum readers don't know the facts, I should first point out that I'm the author of this particular book, so having declared my interest I can now talk about it and you can ignore me if you think it's an abuse of privilege.

Quite a few of you ordered this book from Dabs Press some time ago and are wondering what happened to it, why it's so long in arriving and indeed if it ever will arrive. As I wrote it, I can of course tell you. At the same time it might be interesting to hear how books are produced and eventually reach the reader. It's not obvious without a bit of thought, but there's a great deal more to it than simply writing a book and sending it to a printer.

THE STORY SO FAR...

I finished the final section of the text last September and it was duly forwarded to Dabs for editing and so on, more of which in a moment. In fairness to Dabs I should at this point say that they'd hoped, even expected that I would be able to complete this part of the job much sooner and had based their advertised publication date on that. Unfortunately, like most of you, I still have to earn a living, and as you'll know this often interferes with things you'd rather be doing, like using the 512 (or even talking about it).

Following this, the text was sent for editing, a job carried out by Sid Day, who you may like to know is also (naturally!) a 512 Forum reader (Hi Sid!). You should understand that editing a book the size of the 512 Technical Guide (400+ pages at the last count) is no small or easy task, but even so Sid had completed the job in about a month. Thus, in early November it arrived back at Dabs ready for typesetting, the next stage in the process.

Typesetting is an extremely laborious, time consuming task, and just like writing or editing a book, there's no way to automate the process. As I said, most of us have to earn a living and Dabs is no exception, therefore typsetting has to be 'slotted in' between servicing orders, answering queries, preparing for and attending shows and generally keeping the business running. Of course also in December comes that great stealer of time, the Christmas break, followed almost immediately by another computer show, this time BETT at the Barbican Centre.

This brings us up to date at the time of writing, but with the added complication that changes have taken place at Dabs over the Christmas period which have definitely not assisted production, rather the reverse. The changes themselves are no secret, but I won't turn the Forum into a gossip column in spite of my occasional excursions into story telling (like this one) so you must either learn the details elsewhere or remain in the dark.

I'm presently (January) expecting the typeset proofs of the book to arrive here any day, when my next job will be to check everything and correct the inevitable errors which will have crept in during editing or typesetting. I shall then produce the glossary and finally write the index and number the contents list, but only

when the page numbers are guaranteed not to change again. After that it's back to Dabs for re-setting of my late changes or additions.

Assuming this last check doesn't cause page renumbering, all that remains then is a final check of the finished product. When this is complete it's on to the easy part, which is getting the book printed, bound and on sale. Depending on the printer's workload at the time, this usually means between one and two months to first availability on the book shelves.

There you have the complete story. If you ordered the book some time ago all I can say is be patient a little longer. It will appear in due course, hopefully pretty soon after you read this issue of BEEBUG. In answer to those who specifically queried the point, the book has definitely not been either 'shelved' or cancelled.

MATTERS ARISING

One other specific query about the Technical Guide which is frequently raised is the question of whether or not it includes a 512 memory expansion project. The answer to this is 'Yes, it does'. The reason for mention of this point is that more and more over the last year or so the 512's limited memory has proved a stumbling block to running new software. Those who've recently acquired a 512 are in many ways worse off than the rest of us because they don't have ready access to 'old' software and never had the chance to buy a PC+ either.

The reason for memory shortage being a more common problem these days is simple. While a 'standard' PC or clone is limited to 640K of memory and 256K or 384K was common only a short time ago, PC systems with EMS (Expanded Memory System) are much more the norm nowadays. While the original 8086/8088 processor chips and their equivalents could only address 640K of memory (which, compared with CP/M Z80 based systems seemed huge at the time), later chips aren't so limited. For example the 80286 can manage a theoretical 16Mbytes of memory. The problem of course is, as usual, the software.

It's interesting to note that, while computers are often spoken of as mere boxes, just convenient containers in which to run the all important software, the truth is that in terms of developments this view is absolutely incorrect. Certainly some programs are better than others, and certainly some are genuinely imaginative or innovative, but the ultimate limit to what can be achieved is always governed by the hardware, primarily the processor chip.

The way that developments actually happen is that processor manufacturers such as Intel or Motorola compete with each other to produce new and more sophisticated processor chips. Following the arrival of such a new chip, the PC manufacturers then compete to produce new and better machines, followed by software producers who write more complex and sophisticated software to take advantage of the new machines. So you see, the truth is that software must always trail the hardware in terms of capability. That's why I always smile when I hear some supposedly informed and expert individual saying "The hardware isn't important, it's the software that matters"!

Back to the point. In 1985, as a result of processor advances, the Lotus Development Corporation in conjunction with Intel announced the specifications for EMS. A while later the Microsoft Corporation announced that Microsoft Windows would be developed to take advantage of EMS. To cut a long story short this was produced for MS-DOS version 3 and EMS PCs. To maintain the compatibility with earlier and more limited versions of both DOS and processor, new extra commands were provided by the processor chip to handle EMS and overcome the previous 640K limit, while retaining all the existing instructions.

If you're unfamiliar with how EMS works, the easiest way to understand it is to liken it to the

Beeb's sideways RAM. An area of memory is logically mapped at a fixed location in the addressing range of the machine, but it consists of separate sections or blocks of RAM which can be individually switched (paged) in or out by the processor under software control. In effect, although a PC may still only address 640K of RAM directly, the contents of some parts of that 640K (in a 64K segment known as the page frame) can be swapped with areas of the extended memory in 16K chunks, in effect giving overlaid main memory. To be able to do this though, you need both the correct processor and the correct operating system.

Well written software which can take advantage of EMS while retaining compatibility with earlier systems should check the version of DOS and, if appropriate, the type of machine to see if EMS is available. If not then the program can resort to restricting facilities or file sizes, or overlaying programs or data from disc so that it can still run. Unfortunately though, in some cases it just won't work without EMS.

I've digressed a little as usual, so if you're thinking this isn't much to do with the 512 perhaps I should explain. As processor chips and PCs have expanded their capabilities, applications software has tended to grow in complexity (and memory consumption) accordingly. One implication is that you may upgrade to a later version of a package you already use successfully in the 512 only to find, if you're unlucky, that the new version won't run at all.

Alternatively, if luck is with you, the new package may run well enough, but may also consume a great deal more memory. I've had several letters on this topic and have recently upgraded one of my own applications to find the same thing. The 'old' version of the main program was an 'EXE' file of about 105K, but the new one, (which happily does work perfectly by the way) is all of 226K.

Since an unexpanded 'empty' 512 has about 356K free with DOS Plus 2.1, it's easy to see that the original program left about 250K free, but the new one has just about cut this in half. This needn't cause problems with every package of course, but with mine, for example, one immediate problem is that many of my existing files are now too big to load with the new software. A secondary point is that I can no longer load the on-line help at all unless I haven't yet loaded a file, and using a DOS shell from within the application, for example, to copy a file or to format a disc, is also now a no-go area more often than not. The all too frequently seen response is now 'Not enough memory to load XYZ'.

One possible solution with an application like word processing is to divide files into smaller ones (using the old software) and simply have more of them, but with something like a spreadsheet or a graphics program for example, this is likely to be totally out of the question.

This brings us full circle. Upgrade your software and, even if it runs successfully, it's possible that you'll suddenly become acutely aware of the shortage of memory in the 512. The PC+ has been out of production for over a year now, so apart from occasional second-hand sales that's no help, but as I said earlier, there is a memory expansion project in the book.

POSSIBLE DEVELOPMENTS

Finally, on the same subject there is a possibility of a new ready-made 512 expansion board being produced, but this is a very expensive undertaking and it won't go ahead unless the numbers justify it. Note that THIS IS NOT an advertisement and no further information will be forthcoming just yet even if you ask for it, but if you would be interested just drop me a short note and say so. Your letter might very well increase the chances of this product seeing the light of day. B

Writing a Compiler (Part 4)

David Spencer continues his discussion of writing a simple compiler

We will start this month's article by presenting the complete compiler program. This is given in listing 1. Rather than compiling an expression directly into machine code, the operation is done in a number of stages. The compiler itself generates a textual form of a Basic program containing the assembler instructions for the final machine code. This then has to be converted to a true Basic program, and executed in order to produce the actual machine code.

Before explaining any details about the compiler you should try it for yourself. Start by typing in and saving the program. When you run it you are prompted to enter an expression. This will then be compiled into the textual form of the assembly language program. This is both printed out as it is compiled, and also saved with the filename 'Object'. The command:

```
*EXEC Object
```

will load in the textual form and convert it into a true Basic program, which can then be saved. It is important that you have saved the compiler program before performing this stage, as it will be overwritten otherwise. Finally, running the object program will assemble the machine code and save it with the name 'Code'. Do not, however, attempt to run the machine code at this stage. Incidentally, if the compiler produces the messages *Syntax error* or *Mistake* then this means that there is an error in the entered expression, as described in the previous articles in this series. On the other hand, any errors in the compiler itself will be reported in the normal way including the line numbers at which they occur.

HOW IT WORKS

If you study listing 1 then you should be able to recognise the general structure which was described in last month's article. Added to this are the procedures PROCerror and PROCmatch also described last month, and the lexical analysis function, FNlex, from the first part of this series. The syntax-directed translation methods described in last month's article are incorporated into the routines that make up the syntax analyser. For example, line 1210 handles the unary negation operator. It calls PROCfactor to parse whatever is being negated, and then generates the instruction:

```
JSR negate
```

to perform the negation at runtime.

There are also a number of other procedures which are concerned with code generation. Two of these, PROCopen and PROCclose, generate the standard preamble and postamble that is common to the output program, regardless of the expression being compiled, for example the FOR-NEXT loop needed to implement two-pass assembly. The procedure PROCput is used to output each individual assembler instruction as it is generated. A further procedure, PROCdoconst, generates the instructions necessary to stack a constant during the execution of the compiled code.

If you look at the assembly language program produced, you will see that the actual instructions match the example given last month with the exception that the final JSR which is immediately followed by an RTS is condensed into a single JMP instruction. This so-called *Tail Code Optimisation* is a very simple

form of optimisation which all compilers of any worth would perform as a matter of course.

LINKING AND LIBRARIES

At this stage you can try the compiler, generating first an 'Object' file, and then the 'Code' file. However, if you try to run the machine code it will just crash. This is not because of a mistake in the compiler (hopefully), but rather because the machine code generated by the compiler is not directly executable. There are three reasons for this, and we will examine them in the remainder of this article.

Firstly, and most seriously, the compiled code makes calls to various routines to execute the compiled expression. For example, a routine called *add* is required to perform any additions. Similarly, the routines *stack* and *print* are required to stack the operands and print the result respectively. As it stands, the code produced by the compiler simply doesn't contain these routines. Instead, when the machine code is assembled, the calls to these non-existent routines are assembled into calls to arbitrary addresses. For example, the output produced by the compiler contains the variable declaration:

```
subtract=1
```

As a result, whenever the statement:

```
JSR subtract
```

is assembled it will generate an instruction to call to address location 1. Clearly there is unlikely to be an addition routine at this address.

The second problem is that the code is assembled to run from address location 0, as defined by the setting of the program counter variable P% in the 'Object' file. Memory from address 0 up to &8F is reserved for use by the currently active language, such as Basic. This is therefore not an appropriate place in which to load an executable program.

Thirdly, the load address of the assembled machine code doesn't correspond to the address at which it is designed to run. We have already said that the code is assembled to run from location 0, yet it will by default be loaded at the same location at which it was stored when it was assembled.

What we need is another program that takes the output of the compiler, combines it with a number of pre-written routines, such as the addition routine, relocates the resulting code to execute at a sensible address, and then saves the final code in such a way that it will load at the same address. These tasks are the job of a utility called a *Linker*.

The way a Linker works is to look at the machine code generated by the compiler and identify any calls to non-existent routines. It then locates the necessary routines in the *Runtime Library* which is simply a collection of all routines that may be called by the compiled code. The appropriate routines are extracted from the library and added to the end of the machine code. The Linker then alters all the calls to these routines so that they jump to the correct place, and then relocates the entire result so that it will run from a 'sensible' address. Finally, the complete code is saved so that when reloaded it will be at the correct address for execution.

In the final part of this series next month we will develop the Runtime Library and Linker program.

```
    .  REM > :4.$.Compiler
   20 REM Program Expression Compiler
   30 REM Version B1.0
   40 REM Author  David Spencer
   50 REM BEEBUG  March 1990
   60 REM Program subject to copyright
   70 :
   80 ON ERROR OSCLI "SPOOL":REPORT:PRIN
T " at line ";ERL:END
   90 INPUT "Enter Expression " input$
  100 PROCopen
  110 DIM consts(20)
  120 constcount=0
  130 minusflag=FALSE
  140 tok=FNlex
  150 PROCexpr
  160 PROCmatch(0)
  170 PROCclose
  180 END
  190 :
  200 DEF PROCexpr
  210 PROCterm:PROCexpr2
  220 ENDPROC
  230 :
  240 DEF PROCexpr2
  250 IF tok=ASC"+" PROCmatch(ASC"+"):PR
```

```
OCterm:PROCput("JSR add"):PROCexpr2
  260 IF tok=ASC"-" PROCmatch(ASC"-"):PR
OCterm:PROCput("JSR subtract"):PROCexpr2
  270 ENDPROC
  280 :
  290 DEF PROCterm
  300 PROCfactor:PROCterm2
  310 ENDPROC
  320 :
  330 DEF PROCterm2
  340 IF tok=ASC"*" PROCmatch(ASC"*"):PR
OCfactor:PROCput("JSR multiply"):PROCter
m2
  350 IF tok=ASC"/" PROCmatch(ASC"/"):PR
OCfactor:PROCput("JSR divide"):PROCterm2
  360 ENDPROC
  370 :
  380 DEF PROCfactor
  390 IF tok=1 PROCmatch(1):PROCdoconst(
value):ENDPROC
  400 IF token=ASC"(" PROCmatch(ASC"("):
PROCexpr:PROCmatch(ASC")"):ENDPROC
  410 IF token=ASC"_" PROCmatch(ASC"_"):
PROCfactor:PROCput("JSR negate"):ENDPROC
  420 PROCerror
  430 ENDPROC
  440 :
  450 DEF FNlex
  460 REPEAT
  470 IF LEFT$(input$,1)=" " THEN input$
=MID$(input$,2)
  480 UNTIL LEFT$(input$,1)<>" "
  490 IF input$="" THEN =0
  500 A$=LEFT$(input$,1):IF NOT minusfla
g AND A$="-" THEN token=ASC"_":input$=MI
D$(input$,2):=token
  510 IF INSTR("+-*/",A$) THEN token=ASC
input$:input$=MID$(input$,2):minusflag=F
ALSE:=token
  520 IF INSTR("()",A$) THEN token=ASCin
put$:input$=MID$(input$,2):=token
  530 IF INSTR("0123456789",A$)=0 THEN P
RINT "Mistake":OSCLI "SPOOL":END
  540 value=VAL(input$)
  550 REPEAT
  560 IF INSTR("0123456789",LEFT$(input$
,1)) THEN input$=MID$(input$,2)
  570 UNTIL INSTR("0123456789",LEFT$(inp
ut$,1))=0 OR input$=""
  580 minusflag=TRUE
  590 =1
  600 :
  610 DEF PROCerror
  620 PRINT "Syntax Error"
  630 *SPOOL
  640 END
  650 :
  660 DEF PROCmatch(template)
  670 IF tok<>template THEN PROCerror
  680 tok=FNlex
  690 ENDPROC
  700 :
  710 DEF PROCput(out$)
  720 PRINT SPC(6);out$
  730 ENDPROC
  740 :
  750 DEF PROCdoconst(number)
  760 consts(constcount)=number
  770 constcount=constcount+1
  780 var$="op"+STR$constcount
  790 PROCput("LDX #"+var$+" MOD &100")
  800 PROCput("LDY #"+var$+" DIV &100")
  810 PROCput("JSR stack")
  820 ENDPROC
  830 :
  840 DEF PROCopen
  850 *SPOOL Object
  860 RESTORE
  870 REPEAT READ data$
  880 IF data$<>"" PRINT data$
  890 UNTIL data$=""
  900 ENDPROC
  910 :
  920 DEF PROCclose
  930 PROCput("JMP print")
  940 FOR F%=0 TO constcount-1
  950 PRINT ".op";F%+1;TAB(6);"EQUD ";co
nsts(F%)
  960 NEXT
  970 PRINT "]NEXT"
  980 PRINT "OSCLI ""SAVE Code ""+STR$~c
ode+"" ""+STR$~O%"
  990 *SPOOL
 1000 ENDPROC
 1010 :
 1020 DATA "NEW"
 1030 DATA "AUTO"
 1040 DATA "REM Compiled object code"
 1050 DATA "DIM code 256"
 1060 DATA "add=0"
 1070 DATA "subtract=1"
 1080 DATA "multiply=2"
 1090 DATA "divide=3"
 1100 DATA "negate=4"
 1110 DATA "stack=5"
 1120 DATA "print=6"
 1130 DATA "FOR pass=4 TO 7 STEP 3"
 1140 DATA "P%=0:O%=code"
 1150 DATA "[OPT pass"
 1160 DATA ""
```

Master Display ROM (Part 2)

Dave Gilvarry concludes his article by adding the facility to configure the start-up screen colours on a Master.

As we mentioned in Part 1 (BEEBUG Vol.8 No.8), the Master Display ROM can be extended to allow the start-up screen colours to be configured. This is achieved through the detection of ROM service calls 40 and 41; these indicate an unknown *CONFIGURE or *STATUS command respectively. With Part 2 of the ROM installed you can select any of the screen colours 0-7 for both foreground and background, using *CONFIGURE FGND n, or *CONFIGURE BGND n, and this information is stored in the ROM's private byte in CMOS RAM. These options are added to the configuration list displayed by *CONFIGURE. The colour numbers can be displayed by *STATUS, or by *STATUS FGND and *STATUS BGND, and are accessed on any type of break to set the colours using VDU19. This action has to occur on any type of break because, unlike shadow operation, VDU19 settings are reset even by a simple break. As with all *CONFIGURE commands, *STATUS will show the changes immediately, but they do not become effective until a hard break is actioned.

Listing 2 contains the necessary changes and additions to last month's program. Listing 1 from last month should NOT be renumbered as the new lines are designed to interleave, and some (lines 120, 1110, 1160, 1240, 1270 and 1840) are actually replacements. These alterations can either be made directly by working on the original source program (after first taking a copy!), or entered as a separate program and then saved as an ASCII file (using the Master's EDIT facility, or the *TEXTSAVE command of BEEBUG's Basic Booster ROM, in preference to *SPOOL). Last month's program can then be loaded into memory and the ASCII file EXEC'd in. Running the modified program will then produce an image of the extended ROM.

THE ALTERATIONS IN DETAIL

Line 120 changes the ROM name to MDR2; lines 1110 and 1160 change the version numbers; lines 1240-1260 re-organise the actions on reset; line 1840 takes care of the *STATUS and *CONFIGURE commands; and lines 2090-2150 provide extra HELP information.

Lines 2180-2430 change the screen colours by first using Osbyte 161 to read the mode number and shadow flag from byte 10 of the CMOS RAM. If the mode is 7 or 135, no further action is taken. For other modes, lines 2230-2240 use Osbyte 161 again to read byte 30+n (where n is the current ROM number, obtained from location &F4). This CMOS location is automatically reserved for the Master Display ROM by the MOS, and is used here to store the foreground and background colour numbers.

Lines 2250-2260 separate these and store them in locations &91 and &92. The actual colour changes occur in lines 2280-2410 where VDU19,n,m,0,0,0 commands are generated. However, if the foreground and background colours have inadvertently been set to the same value, the foreground colour is first changed (line 2340) to be the complement of the background colour and the new numbers are stored in the CMOS byte (lines 2350-2360).

Line 2450 checks for ROM service call 41 (unknown *STATUS command); line 2470 gets the colour byte as above; and lines 2480-2500, 2750-2810, and 2840-2900 check to see if either 'FGND' or 'BGND' follows the *STATUS command. If either of these has been included, lines 2690-2710 and 2720-2740 display the colour number. If *STATUS has no parameters, lines 2530-2660 extend the normal full status display to show both foreground and background information, while unrecognised parameters result in a 'bad command' error.

If service call 41 is not detected, then line 2930 checks for service call 40 (unknown *CONFIGURE command) and line 2950 checks

the rest of the command line. Unrecognised text gives 'bad command'; no text at all extends the normal full configure display to show the syntax of the new *CONFIGURE commands (lines 3000-3080); and 'FGND' or 'BGND', followed by an optional space and a single decimal digit, causes the corresponding colour number to be set in lines 2960-2970 (if the colour number is omitted from the command, a 'bad command' error again results). Finally, line 3100 (foreground) or 3150 (background) reads the private byte once more; lines 3220-3240 (foreground), or 3300-3320 (background), change the appropriate part of the byte, and lines 3250-3270 write the new byte back to CMOS RAM using Osbyte 162.

Listing 2

```
  120 *SAVE MDR2 5000+400 8000 8000
 1110 EQUB2
 1160 EQUS" 1.2v"
 1240 CMP#&27:BNEnotbreak:JMPbreak
 1250 .notbreak
 1260 JMPtryhelp
 1270 .duncol
 1840 JMPtrystatus
 2090 EQUS"  Extended STATUS & CONFIGURE
"
 2100 EQUB13
 2120 EQUB13
 2130 EQUS"  Foreground via *CO. FGND n"
 2140 EQUB13
 2150 EQUS"  Background via *CO. BGND n"
 2180 .break
 2190 LDX#10:LDY#0:LDA#161:JSRosbyte
 2200 TYA:STA&90:AND#7:CMP#7:BEQseven
 2210 JSRsbyte:JMPvdu
 2220 .sbyte
 2230 CLC:LDAromnum:ADC#30:TAX
 2240 LDA#161:JSRosbyte
 2250 TYA:AND#7:STA&91
 2260 TYA:AND#112:LSRA:LSRA:LSRA:LSRA:ST
A&92
 2270 RTS
 2280 .vdu
 2290 LDA#19:JSRoswrch
 2300 LDA#0:JSRoswrch
 2310 LDA&92:JSRoswrch:LDA#0:JSRoswrch
 2320 JSRoswrch:JSRoswrch
 2330 LDA&91:CMP&92:BNEdifferent
 2340 EOR#7:STA&91:LDA&92:ASLA:ASLA:ASLA
:ASLA:ORA&91
 2350 TAY:CLC:LDAromnum:ADC#30:TAX
 2360 LDA#162:JSRosbyte
 2370 .different
 2380 LDA#19:JSRoswrch
 2390 LDA#7:JSRoswrch
 2400 LDA&91:JSRoswrch:LDA#0:JSRoswrch
 2410 JSRoswrch:JSRoswrch
 2420 .seven
 2430 JMPduncol
 2440 .trystatus
 2450 CMP#41:BEQstatus:JMPtryconfig
 2460 .status
 2470 PHY:JSRsbyte:PLY
 2480 LDA(comline),Y:CMP#13:BEQpstatus
 2490 JSRtryf:BCCsfgnd
 2500 JSRtryb:BCCsbgnd
 2510 JMPrestore
 2520 .pstatus
 2530 LDX#0
 2540 .pfgnd
 2550 LDAfmess,X:JSRosasci
 2560 INX:CPX#9:BCCpfgnd
 2570 LDA&91:ORA#48:JSRosasci:JSRosnewl
 2580 LDX#0:JMPpbgnd
 2590 .fmess
 2600 EQUS"FGND      <n> (n = 0-7)"
 2610 LDX#0
 2620 .pbgnd
 2630 LDAbmess,X:JSRosasci
 2640 INX:CPX#9:BCCpbgnd
 2650 LDA&92:ORA#48:JSRosasci:JSRosnewl
 2660 JMPrestore
 2670 .bmess
 2680 EQUS"BGND      <n> (n = 0-7)"
 2690 .sfgnd
 2700 LDA&91:ORA#48:JSRosasci:JSRosnewl
 2710 JMPalldone
 2720 .sbgnd
 2730 LDA&92:ORA#48:JSRosasci:JSRosnewl
 2740 JMPalldone
 2750 .tryf
 2760 DEY:LDX#255
 2770 .loopf
 2780 INX:INY
 2790 CPX#4:BEQsok
 2800 LDA(comline),Y:AND#&DF:CMPfmess,X:
BEQloopf
 2810 JMPfault
 2820 .sok
 2830 CLC:RTS
 2840 .tryb
 2850 DEY:LDX#255
 2860 .loopb
 2870 INX:INY
 2880 CPX#4:BEQsok
 2890 LDA(comline),Y:AND#&DF:CMPbmess,X:
```

```
BEQloopb
 2900 .fault
 2910 SEC:RTS
 2920 .tryconfig
 2930 CMP#40:BEQconf:JMPrestore
 2940 .conf
 2950 LDA(comline),Y:CMP#13:BEQpconf
 2960 JSRtryf:BCCcfgnd
 2970 JSRtryb:BCCcbgnd
 2980 JMPrestore
 2990 .pconf
 3000 LDX#0
 3010 .pcf
 3020 LDAfmess,X:JSRosasci
 3030 INX:CPX#23:BCCpcf
 3040 JSRosnewl:LDX#0
 3050 .pcb
 3060 LDAbmess,X:JSRosasci
 3070 INX:CPX#23:BCCpcb
 3080 JSRosnewl:JMPrestore
 3090 .cfgnd
 3100 PHY:JSRsbyte:PLY:JSRspaces
 3110 JSRfbyte:BCCcdone:JMPrestore
 3120 .cdone
 3130 JMPalldone
 3140 .cbgnd
 3150 PHY:JSRsbyte:PLY:JSRspaces
 3160 JSRbbyte:BCCcdone:JMPrestore
 3170 .spaces
 3180 LDA(comline),Y:CMP#32:BNErtn
 3190 INY:BRAspaces
 3200 .rtn
 3210 RTS
 3220 .fbyte
 3230 CMP#13:BEQnonum:AND#7
 3240 STA&91:LDA&92:ASLA:ASLA:ASLA:ASLA:
ORA&91
 3250 .setconf
 3260 TAY:CLC:LDAromnum:ADC#30:TAX
 3270 LDA#162:JSRosbyte:CLC:RTS
 3280 .nonum
 3290 SEC:RTS
 3300 .bbyte
 3310 CMP#13:BEQnonum:AND#7
 3320 STA&92:ASLA:ASLA:ASLA:ASLA:ORA&91
 3330 JMPsetconf
```

Printing Characters 128 to 255 (continued from page 33)

```
 1640 DATA 6600666666663C00,7EC39DB19DC37E00
 1650 DATA 0018387F38180000,00181CFE1C180000
 1660 DATA 181818187E3C1800,00183C7E18181818
 1670 DATA 30183C063E663E00,30183C667E603C00
 1680 DATA 66003C667E603C00,3C663C667E603C00
 1690 DATA 66003C063E663E00,3C663C063E663E00
 1700 DATA 00003F0D3F6C3F00,00003C6660663C60
 1710 DATA 0C183C667E603C00,66003C6666663C00
 1720 DATA 6600666666663E00,3018003818183C00
 1730 DATA 3C66003818183C00,3018003C66663C00
 1740 DATA 3C66003C66663C00,3018006666663E00
 1750 DATA 3C66006666663E00,66006666663E063C
 1760 DATA 00663C66663C6600,3C603C663C063C00
 1770 DATA 3C663C0000000000,0000001818181818
 1780 DATA 0000001F00000000,0000001F18181818
 1790 DATA 000000F800000000,000000F818181818
 1800 DATA 000000FF00000000,000000FF18181818
 1810 DATA 1818181800000000,1818181818181818
 1820 DATA 1818181F00000000,1818181F18181818
 1830 DATA 181818F800000000,181818F818181818
 1840 DATA 181818FF00000000,181818FF18181818
 1850 DATA 000000070C181818,000000E030181818
 1860 DATA 18180C0700000000,18183E0000000000
 1870 DATA 1800181830663C00,1800181818181800
 1880 DATA 366C0066766E6600,366C007C66666600
 1890 DATA 187E181818181800,187E1818187E1800
 1900 DATA 1818180000000000,30180C0000000000
 1910 DATA 3F7B7B3B1B1B1F00,0000001818000000
 1920 DATA 03030606761C0C00,AA55AA55AA55AA55
 1930 DATA 3E63676B73633E00,1C3663637F636300
 1940 DATA 7E33333E33337E00,7F63606060606000
 1950 DATA 1C1C363663637F00,7F33303E30337F00
 1960 DATA 7E660C1830667E00,7733333F33337700
 1970 DATA 3E63637F63633E00,3C18181818183C00
 1980 DATA 63666C786C666300,1C1C363663636300
 1990 DATA 63777F6B63636300,63737B6F67636300
 2000 DATA 7E00003C00007E00,3E63636363633E00
 2010 DATA 7F36363636363600,7E33333E30307800
 2020 DATA 7F63301830637F00,7E5A1818181818C0
 2030 DATA 6666663C18183C00,3E083E6B3E083E00
 2040 DATA 6363361C36636300,3E086B6B3E083E00
 2050 DATA 3E63636336366300,7F636336361C1C00
 2060 DATA 18187E1818007E00,007E0018187E1818
 2070 DATA 1818181818181800,3636363636363600
 2080 DATA 0066666666663C00,003C666666666600
 2090 DATA 00033E676B733E60,00003B6E666E3B00
 2100 DATA 1E33333E33333E60,000066361C183030
 2110 DATA 3C60303C66663C00,00001E301C301E00
 2120 DATA 3E0C183060603E06,00007C6666660606
 2130 DATA 3C66667E66663C00,0000181818180C00
 2140 DATA 0000666C786C6600,6030181C36636300
 2150 DATA 0000333333333E60,000063331B1E1C00
 2160 DATA 3C60603C60603E06,00003E6363633E00
 2170 DATA 00007F3636363600,00003C66667C6060
 2180 DATA 00003F6666663C00,00007E1818180C00
 2190 DATA 0000733333331E00,00003E6B6B3E1818
 2200 DATA 000066361C1C3633,0000636B6B3E1818
 2210 DATA 000036636B7F3600,380C063E66663C00
 2220 DATA 00316B46007F0000,007E007E007E0000
 2230 DATA 071C701C07007F00,060C7E187E306000
 2240 DATA 701C071C70007F00,FFFFFFFFFFFFFFFF
```

Cable Analyser Review

Alan Wrigley reviews the CA100 cable analyser from Electronic Innovations Ltd

The BBC micro, with its multitude of output ports, has long been considered an ideal micro for applications which involve an interface with the outside world. As a result, many laboratories and workshops have BBCs busily beavering away connected to various items of test or control equipment.

The CA100 cable analyser is the latest piece of test equipment to appear on the market. It may be of interest to anyone who designs, manufactures or services cables in quantity, and requires a quick but comprehensive means of testing them. It can also be used with an Archimedes equipped with a user port podule.

The prototype CA100 analyser

THE PRODUCT

The prototype analyser supplied for review was housed in a sturdy metal box, 250x180x70mm, and appeared to be well constructed. It is connected to the computer via a flying ribbon cable which plugs into the BBC's user port. No external power connection is necessary. On the front of the box are four 25-way 'D' connectors, into which the cable to be analysed may be plugged. A special interface PCB will be separately available, which will plug into the four 'D' plugs and will be drilled to accommodate a wide variety of common connectors, allowing the user to solder items to the board to suit his own requirements.

Supplied with the analyser is a software package which provides a comprehensive range of functions, to be described below. The documentation provided with the prototype was sparse, but full documentation will be supplied with the production units.

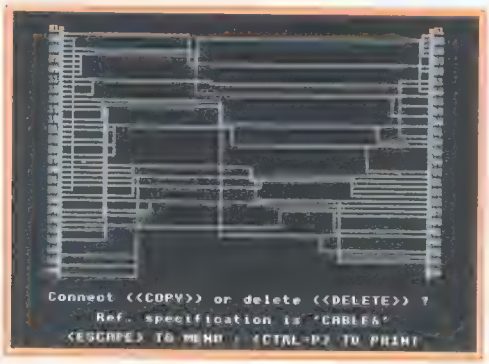
THE SOFTWARE

Booting up the disc reveals the main menu, containing 9 options. One of these allows you to change various default settings, namely colour or monochrome display, printer type, and serial or parallel output. You can set these so that the options you require become the defaults in future. The other menu selections allow you to manipulate cable data in various ways, as will now be explained.

Each time a cable is analysed, a set of data will be produced, known as the *cableform*. This basically indicates which connectors at one end of the cable are connected to which ones at the other. All the software functions operate on one cableform at a time, which is held in memory. This is always referred to as the "reference specification", and may have been derived from an actual cable under test, from a cableform file held on disc, or entered manually using the editor to be described later. The reference specification may also be saved to disc.

Once a cableform is in memory, an option from the main menu allows you to list the pin connections on screen, or to a printer. A further option compares the data held in memory to a cable currently plugged into the analyser and displays any deviations between the two, again

with the option of a printout. Thirdly, you can display the cableform graphically on the screen. This shows all 100 pins, in two columns of 50 at each side of the screen, with connecting lines drawn where pins are connected by the cable.



A cableform displayed on the screen

All these functions appear to work efficiently. I was a little disappointed to find that setting the default screen option to colour had no effect on the cableform graphic display, which is simply white on black (both on the model B and the Master). This made it rather difficult to read a complicated diagram, with many lines packed closely together, on a medium-resolution monitor.

THE DESIGN MENU

The functions described so far are intended to provide simple test operations which enable cable analysis to be carried out quickly and conveniently, if necessary by non-experts. For the more experienced user, a higher level of sophistication is provided by a Design Menu, which can be entered from the main menu.

The design programs give you more control over the analysis of the test cable, allowing you to specify the number and size of the connectors, which are then differentiated on the graphic display. The display diagram may be dumped to a printer, and also a cableform may be edited manually, pin by pin. Unfortunately, there were a number of problems with the design suite software which made it difficult to assess their worth. I was assured by the manufacturers that these would be sorted out before full production started.

CONCLUSIONS

The analyser itself is well made, giving the appearance of being constructed to a specification rather than a price. The basic software functions are adequate for their task and would be useful in situations where standard cables need testing quickly. Provided that the design programs are overhauled and are able to do what is claimed for them, the CA100 could be a useful tool, being certainly a lot easier and faster to use than a continuity tester. However, I feel that the price will deter all except those who simply cannot manage without its facilities.

Points Arising....Points Arising....Points Arising....Points Arising...

CROSSWORD EDITOR (BEEBUG Vol.7 No.4)

This does not correctly display a file longer than 256 bytes (17 by 17) when reloaded. This can be corrected by modifying the following three lines as shown:

```
 2040 PRINTTAB(26,0)mode$;" MODE":!&72=W
%:CALL&A00
 3350 LDA#&FF:STA&71:LDX#0:DEC&73
 3370 TYA:PHA:LDY&71:CPY#0:BNE w:INC&73:
.w LDA (&72),Y
```

Thanks to Joe Hakeney for discovering and correcting this bug.

DISC FILE IDENTIFIER (BEEBUG Vol.8 No.7)

Line 6310 should be amended to read:

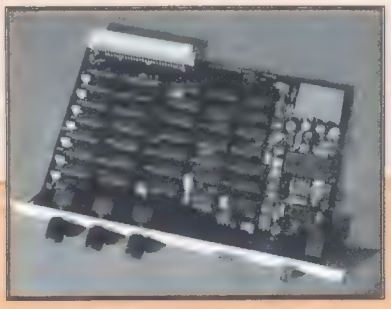
```
UNTIL data=&0D OR data=&FF OR LEN(m$)=
255
```

to improve file type recognition.

RISC USER

The Archimedes Magazine & Support Group

Risc User is enjoying the largest circulation of any magazine devoted solely to the Archimedes range of computers. Now in its third year of publication, it provides support to schools, colleges, universities, industry, government establishments and private individuals. Existing Beebug members, interested in the new range of Acorn micros, may either transfer their membership to the new magazine or extend their subscription to include both magazines.
A joint subscription will enable you to keep completely up-to-date with all innovations and the latest information from Acorn and other suppliers on the complete range of BBC micros. RISC User has a massive amount to offer to enthusiasts and professionals at all levels.



Here are just some of the topics covered in more recent issues of RISC User:

A WIMP FRONT END FOR THE BASIC EDITOR
A utility which provides a WIMP Desktop front end to the Basic Editor.

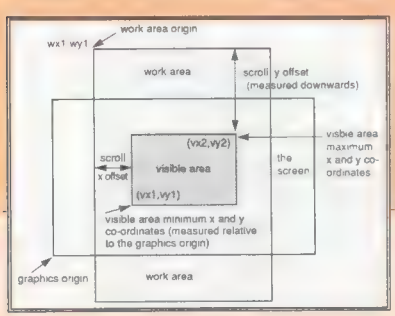
PINEAPPLE'S COLOUR DIGITISER
A review of the new real-time colour digitiser for the Archimedes.

BITS AND PCS
Tips provided for the users of the PC emulator.

A utility which creates additional screen modes for the Archimedes in the form of relocatable modules.

UNDER THE LID
A major series explaining the hardware that makes up the Archimedes.

POP-UP MENUS
Add pop-up menus to non-Wimp programs



ASSEMBLER WORKSHOP
A major series for the more advanced ARM processor programmer.

How to use the Draw application for simple desktop publishing.

INTRODUCING LASER PRINTERS
A look at the workings of laser printers and Page Description Languages.

MASTERING THE WIMP
A major series for beginners to the WIMP programming environment. This month - Forced window redraws,

SCSI INTERFACES
What is a SCSI interface ? A comparative review of SCSI interfaces for the Archimedes and A3000.

PROBING PASCAL
A brief overview of the Pascal language and the two compilers available for the Archimedes.

As a member of BEEBUG you may extend your subscription to include RISC User for only £8.10 (overseas see below).

Don't delay!

Phone your instructions now on (0727) 40303

Or, send your cheque/postal order to the address below. Please quote your **name** and **membership number.** When ordering by Access, Visa or Connect, please quote your card number and the expiry date.

SUBSCRIPTION DETAILS

Destination	Additional Cost
UK,BFPO &Ch Is	£ 8.10
Rest of Europe and Eire	£12.00
Middle East	£14.00
Americas and Africa	£15.00
Elsewhere	£17.00

RISC User, 117 Hatfield Road, St Albans, Herts AL1 4JS, Telephone (0727) 40303, FAX (0727) 60263

BEEBUG Education

by Mark K. Sealey

The whole of this month's Beebug Education is unashamedly devoted to a superb program called Printbox from market leaders Logotron. Recently merged with Longman, Logotron is already renowned for both BBC and Archimedes versions of LOGO, Pendown and associated utilities, and Hyperbook as well as the new and innovative Numerator. Printbox should prove useful with children and students over a wide range of ages.

Just as you might have thought that good, child-centred software for the BBC micro was on the way out, Printbox appears to fill a niche in the market in a way which ought to make teachers, presenters and lecturers, as well as other computer enthusiasts, take notice.

Printbox comes either on ROM together with a disc (for the BBC model B), or on two discs (5.25" or 3.5") for the Master 128 and Master Compact machines. It enables you to produce quality artwork in single page format with the ability to design and edit layouts, incorporate clip-art and use up to 16 fonts.

BASIC MENU AND STARTUP

The initial menu is well laid out. It offers you the chance to change the working colours, though it is easy to do this later on, and allows access to Operating System (*) commands as well as QUITting. The other options are ones to edit an old page or begin a new page, browse, and edit and load fonts as well as clip art.

This is a sensible arrangement, and the progress to other menus in what is quite a sophisticated functions structure, is as near ideal as can be. There is one small exception when several fonts, for instance, have been loaded. In order to leave as much working space on the screen as possible, just one line of text handles the selection of these fonts. It soon becomes necessary to scroll sideways, when a right angle-bracket (>), used otherwise at times for prompts, is not as intuitive as it might be. This is a small point, though, and recourse to the excellent 80 page manual will explain all.

DOCUMENTATION

Apart from the main manual, the documentation comprises not only a function key card, a diagram of the menu structure and quick reference guide, but also a second, exemplary, ideas book with all the sample fonts and clip-art library for users to choose styles - although it is possible to cycle through sections of these as stored on disc. Given the price of the package, these manuals alone should convince potential users that they have a bargain here.

USE OF PRINTBOX

At the heart of operations is a "dotty box", really a scalable window which is extremely easy to use. It is controlled by the cursor keys, and surrounds the chunk of material you want to work on. You can then copy, flip, move, invert or even shrink/enlarge any piece of text to achieve just the right effect.

Because this is so simple to manipulate, the package is less one to show off the power of a micro or the tricks of the programmer, but more one to encourage the user to try and retry a variety of effects until just what they want is on screen. Even the most user-friendly full blown DTP packages sometimes fall down here!

Once you are satisfied with what is in front of you, it can be saved to a second data disc much as you might expect. You can then proceed to quit and/or print out your page. Once again, Logotron has taken into account what is actually available in schools in terms of

printers; dumps are provided not only for Epson and Integrex but also NLQ, 24 pin and Canon setups.

Sample printout using Printbox

In terms of the aesthetics of the finished product, Printbox has a lot going for it. Effects including shadow, different backgrounds and borders are all excellent and easy to produce. One result of this - and a pointer to the overall flexibility of the product - is that mathematically inspired graphics (maybe the sort of which turtle graphics is capable) can be blended very effectively with suitable text, both descriptive and explanatory.

OTHER OPTIONS

A further feature of Printbox is that, quite independently of other options, screens (from LOGO, for example) can be grabbed to be incorporated in Printbox artwork. One of the sub-menus from the Edit Page (which you will come to visit quite frequently) allows these to be re-incorporated at any stage. Once again, the manual is very clear in guiding you through this process.

CLIP-ART

There is a clip-art editor which will be all the more fun to use since the 'icons' produced may later find their way into the pupils' work. Even so, the quality of what is supplied as samples is high. There are seven items relating to Christmas, for instance, twenty "Celtic" patterns and some 100 small pictures in all. Not only will these satisfy many less ambitious pupils for some time, but also inspire more confident ones to work at their own.

Into the bargain, the "Glue" utility, also on disc, allows you to manage and manipulate sprites - and rescale oversize ones. It is also envisaged that Printbox will be used in conjunction with word processors like Pendown, from which it is easy to incorporate text files; Printbox can also cope with text flow from one box to another.

It is hardly surprising that with all these features and the need, presumably, for buffers, memory can be tight. There is a memory report option in the main menu, which helps monitor this. At the same time, users would be advised to proceed cautiously until familiar with just what can be fitted in with various art/font/text combinations.

Error reporting is more than adequate and also well covered in the manual. There is also a useful section describing why and how to build boot files to load just the font and clip-art files that are needed for any one project. This is a nice touch since it will make life easier whenever it is appropriate to have the right files from many loaded automatically.

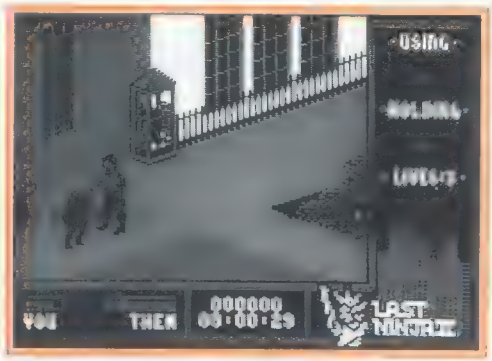
CONCLUSIONS

In practice then, this is software which can be approached and learnt very easily. It sports nearly all the features you would want in a package of its kind. It works quickly and is straightforward yet capable of sophisticated and exacting results. Not only could infants get to grips with computer "presentation" easily and painlessly, but computer literacy or secondary/FE art departments could make imaginative and stylish use of it.

Given the features and the accessibility - thanks to sample files and documentation, for instance Printbox can be wholeheartedly recommended, and represents uncommonly good value for money.

to collect the various keys located around the screens. There are moving objects to avoid and puzzles to solve in achieving your task. The graphics and sound are both good, and key control is kept very simple. It is easy to play in the early stages and quickly becomes quite addictive.

To sum up: this is one of the better collections Superior has released in this series. The two combat games are good, provided they are your type of game. The other two are probably of wider appeal, and definitely have that addictive feel.



Last Ninja 2

The other new release from Superior is *The Last Ninja 2*. This is the sequel to the original game, as featured on Play It Again Sam 12. The scenario has moved on in Ninja 2 from the Dark Ages in the Far East, to the 1990s in downtown New York. Here your adversaries are somewhat different, but are still just as difficult to overcome. There are various levels to work through in your quest, from the paths of Central Park alive with muggers, to the streets of Manhattan with its down and outs.

Graphics and game play in Ninja 2 are very similar to the original. If you enjoyed the first game then you will probably like this one, too. For myself, I found the game enjoyable but the key combinations, as in the original, sometimes left me in a state of some confusion with the inevitable loss of a life. Don't let this put you off though.

Points Arising....Points Arising....Points Arising....Points Arising....

EDIKIT
(BEEBUG Vol.8 Nos.7-9)

The articles and programs which make up the EdiKit ROM refer to RAM bank 'X'. A RAM bank will normally be a number in the range 0 to 7 (certainly on a Master), but may take other values (or letters) depending on make. Refer to the manufacturer's instructions, and use an appropriate value (i.e. that of a vacant RAM bank) in place of the 'X' where indicated in the programs.

We apologise for the missing lines of text in last month's article in this series. However, all the relevant information also appeared elsewhere in the article.

MINEFIELD (BEEBUG Vol.8 No.8)

Owing to some last-minute amendments there are a few errors in the listing, for which we apologise. The following three lines should read:

```
100 ON ERROR GOTO230
2270 G%=&8C0:FOR A%=0 TO 9:READ A$:C%=0
2300 NEXT:READ A$:FOR A%=0 TO 3:VDU 23,2
24+A%:FOR B%=0 TO 7:VDU EVAL("&"+MID$(A$,
A%*16+B%*2+1,2)):NEXT,:ENDPROC
```

BEEBUG MAGAZINE DISC
(Vol.8 No.8)

The menu program may fail to work correctly on a model B or B+ because of insufficient memory. To correct this, load the menu program using LOAD"Menu" and amend line 70 to read:

```
70 DIM P% 3000    (not 5000)
```

and resave using SAVE"Menu".

The Magscan file on this disc (Vol.8 No.8) was incorrectly formatted. A corrected version is included on this month's disc along with the Magscan file for this issue (Vol.8 No.9).

Hopefully, this clears up any outstanding problems.

MORE MUGS

I was glad to see an article on MUGs in the December issue (Vol.8 No.7). However, I think that the statement that "in all games there is a certain amount of violence" is incorrect and misleading. While it is true that some games such as *Mirrorworld* and *Shades* do encourage 'fighting', there is no fighting at all in *The Zone*. Indeed, the aim there is more to co-operate and be friendly with the mobiles and other players. Similarly, in *Gods*, there is very little fighting and much to be gained by making friends with the other players.

I would prefer to emphasize the fun of role-playing in MUGs. Many players derive much enjoyment from assuming a 'persona' and acting out the new character in conversation and interaction with others in the game. There have been many famous and successful role-players in the games I have been associated with, whether they be men pretending to be women, people pretending to be animals, someone playing a ghost or whatever.

The 'full' list of MUGs failed to mention *The Zone* (01-994 9119), *Trash* (soon to be on Prestel, or *Shades* (Prestel and 0342 810905). There are probably others. As we saw at Adventure Convention, there are plenty of new MUGs under development, with something to suit anyone adventurous.

Sheila Thomas (Malwen & Isolde)

ALTERNATIVE PLUS-OR-MINUS

I would like to suggest an alternative for creating a Plus-or-Minus sign to that described by Sebastian Lazareno in BEEBUG Vol.8 No.6. This is based on a method I have employed for some time to cause my Epson compatible Panasonic KX-P1081 printer to match the '£' and '#' keys on the keyboard. In the program, the data at lines 180 and 190 define these last two characters. I have chosen the backslash '\' (CHR$92) to redefine as a @ in both the computer and the printer (lines 160 and 200 respectively).

```
100 MODE 4:VDU2
110 FOR I%=1 TO 3:VDU1,27,1,121
120 FOR J%=1 TO 10
130 READ C%:VDU1,C%
140 VDU3
150 NEXT J%,I%
160 VDU23,92,8,8,127,8,8,0,127,0
170 END
200 DATA 35,40,0,254,0,40,0,254,0,40
210 DATA 96,0,2,16,14,122,130,16,130,68
220 DATA 92,0,0,34,34,34,250,34,34,34
```

George Stenning

Thanks to Mr.Stenning for this alternative suggestion. However, despite the fact that the Panasonic printer referred to is often claimed to be Epson compatible, in the above case this is not true. We give below a version of the program which has been tested on an Epson FX80:

```
100 VDU2,1,27,1,58,1,0,1,0,1,0
110 FOR I%=1 TO 3:VDU1,27,1,38,1,0
120 READ A%:VDU1,A%,1,A%,1,139
130 FOR J%=1 TO 11:READ A%:VDU1,A%:NEXT
140 NEXT
150 VDU1,27,1,37,1,1,1,0,1,35,1,96,1,92,1,
13,1,10,1,27,1,37,1,0,1,0,3
160 DATA35,40,0,254,0,40,0,254,0,40,0,0
170 DATA96,0,2,16,14,122,130,16,130,68,0,0
180 DATA92,0,0,34,34,34,250,34,34,34,0,0
```

The Epson requires more codes, as it needs to be informed beforehand when characters are to be downloaded, and the existing character definitions must be accessed in advance. In addition, character definitions require 11 bytes not 9. Line 150 prints out the redefined characters.

UNDERLINING THE POINT

I am not over familiar with the operations of a computer. Recently when typing in some listings from the magazine, there has been a small line as shown in the following examples:

```
PROCcontrol_it
BCC print_it
```

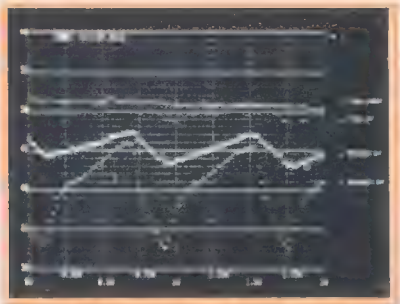
Can you please explain what this line means, and how I show or type it when entering the program.

L.W.Bates

*The character referred to is the **underline** or **underscore** character which appears on the keyboard to the immediate left of the cursor keys and on the same key as the pound sign. It has no particular programming meaning, but is one of the very few characters acceptable in procedure and variable names in addition to the upper and lower case alphabet and the digits. It therefore makes a useful spacer for programmers wanting to use two (or more) word names as in the examples.*

Another character which can cause similar problems, particularly to newcomers, is the vertical bar character (|), again to the immediate left of the cursor keys and on the same key as the backslash (\).

The Best of BEEBUG



Basic Booster ROM

Super Squeeze A program compressor.

Partial Renumber A very useful utility which renumbers a selected block of lines.

Program Lister List any program direct from a file.

Resequencer Rearrange the lines in a Basic program - line numbering is automatically adjusted.

Smart Renumber Renumber a program so that procedures start at a particular line number.

Textload and Textsave Save and load a Basic program as text.

Applications II Disc

Share Investor
A program which assists decision making when buying and selling stocks and shares.

Real Time Clock
A real time digital alarm clock displayed for all BBC micros.

Running Four Temperatures
A program for calibrating and plotting up to four temperatures

Crossword Editor
Design, edit and solve crosswords with this program.

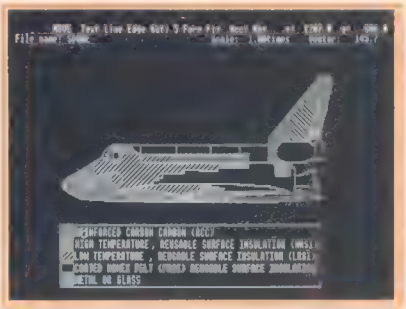
Label Processor
Design, save and print labels at any size on an Epson compatible printer.

Monthly Desk Diary
A month-to-view calendar which can be used on-screen or printed out.

3D Landscapes
Create computer-generated three dimensional landscapes.

Foreign Language Tester
Define foreign characters and test your knowledge of foreign languages

Julia Sets
Fascinating displays of Julia sets, the extensions of the Mandelbrot set.



ASTAAD

Master

- full mouse and joystick control
- built-in printer dump
- speed improvement
- STEAMS image manipulator
- Keystrips for ASTAAD and STEAMS
- Comprehensive user guide
- Sample picture files

General Utilities Disc

- Printer Buffer *
- Sprite Editor/Animator
- Multi-Character Printer Driver for View
- Mode 7 Screen Editor
- Multi-Column Printing
- Epson Character Definer
- ROM Controller
- ROM Filing System Generator
- BEEBUG MiniWimp †

* Master series only. † Requires sideways RAM.

Applications I Disc

- Business Graphics
- Video Cataloguer
- World by Night and Day
- Phone Book
- Page Designer
- Personalised Letter-Heads
- Mapping the British Isles
- Selective Breeding
- Appointments Diary
- The Earth from Space
- Personalised Address Book

(ASTAAD disc - members price of £9.95 , non-members price £19.95)

Applications II *Code 1411A (80 track DFS)*	❑	Applications II *Code 1412A (3.5" ADFS)*	❑
Basic Booster ROM *Code 1403A*	❑	Basic Booster Disc *Code 1402A*	❑
ASTAAD Code 1407A *(80 track DFS)*	❑	ASTAAD Code 1408A *(3.5" ADFS)*	❑
General Utilities Disc *Code 1405A (80 track DFS)*	❑	General Utilities Disc *Code 1413A (3.5" ADFS)*	❑
Applications I Disc *Code 1404A (80 track DFS)*	❑	Applications I Disc *Code 1409A (3.5" ADFS)*	❑

Please add p&p - 60p for the first item and 30p for every additional item.

Phone your order now on (0727) 40303

or send your cheque/postal order to the address below. Please quote your **name** and **membership number.** When ordering by Access, Visa or Connect, please quote your card number and the expiry date.

BEEBUG Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Telephone (0727) 40303.

HARD COPY CATALOGUE IN WORDWISE

Tom Boyd

The following short program creates a machine code utility which allows hard copy of any DFS disc catalogue to be produced while using Wordwise. Running the program saves the code with the name 'PD'. Provided this is on the relevant disc(s) typing *PD will then print out a copy of the contents of the current DFS disc. The code is located at &900 (see lines 40 and 190), but this could be changed.

```
 10 REM PRINT DIRECTORY
 20 REM Author T.K.Boyd
 30 :
 40 MC=&900
 50 FOR C1=0 TO 3 STEP 3
 60 P%=MC
 70 [OPT C1
 80 LDA #2:JSR &FFEE
 90 LDX #MSG MOD 256
100 LDY #MSG DIV 256
110 JSR &FFF7
120 LDA #3:JSR &FFEE
130 RTS
140 .MSG NOP
150 ]
160 NEXT
170 :
180 A$="*."+CHR$(13)
190 $(P%-1)=A$:L%=P%+LEN(A$)-&900
200 C$="*SA.PD 900 +"+STR$~(L%)
210 OSCLI C$
220 END
```

PROTECTING VIEW FILES ON A MASTER

David Holton

Although encryption programs usually work by EORing a random byte value with each byte of a file, a much simpler way exists for preventing other users from seeing the contents of your files.

View expects to find a carriage return (&0D) at the end of a file, and if it doesn't, it gives "NO TEXT" and won't change to the editing screen. Thus to *lock* a file, use *APPEND to put a byte other than &0D on the end. Type:

```
*AP. <filename>
```

and when the figure '1' appears, type any character(s) you like. I always use '*' (CHR$42), because 42 is the answer to life, the universe and everything! Do NOT press Return, but Escape. Now load the file and try to screen it.

There is no easy way to remove the unwanted character(s) when you want to unlock the file. So use *APPEND again, but press Return in response to the '1', and then Escape in response to the following '2'. This places an &0D character at the end of the file. Load the file into View and edit out the unwanted characters.

You can, of course, use *TYPE to see what's in a NO TEXT file, but it should keep the uninitiated out of your files.

CONVERTING A NIBBLE TO A BYTE

Kai Ng

The following routine may be of interest to machine code programmers by providing an exceedingly compact method of converting a binary nibble into a hexadecimal ASCII character.

```
.mshex  LSR A:LSR A:LSR A:LSR A
.lshex  PHP:SED
        AND #&0F:CMP #&0A:ADC #&30
        PLP:RTS
```

Thus *JSR lshex* would apply conversion to the lower 4-bit nibble of the accumulator register, while *JSR mshex* would convert the upper nibble.

QUICK CIRCLE

John McFarlane

Filled circles can be tedious and time-consuming to draw using sines. The following procedure uses Pythagoras' theory which relates the squares of the lengths of the sides of a right-angle triangle. It will work in any graphics mode. X% and Y% are the co-ordinates of the centre, and R% the radius. Changing the value of Z from 1 alters the width of the circle to form an ellipse.

```
 100 MODE 4
 110 PROCcircle(100,640,512,2)
 120 END
 130 :
1000 DEF PROCcircle(R%,X%,Y%,Z)
1010 LOCAL A%,B%,S%
1020 S%=R%*R%
1030 FOR A%=-R% TO R% STEP 4
1040 B%=SQR(S%-A%*A%)*Z
1050 MOVE X%-B%+4,Y%+A%:DRAW X%+B%,Y%+A%
1060 NEXT:ENDPROC
```

Personal Ads

copies including documentation to avoid any abuse of this facility.

WANTED: Would like some assistance with Printwise 40T DFS using preferably Interword. **FOR SALE:** Watford sideways ROM/RAM board with battery backed 16k sideways RAM (not fitted) and 32k sideways RAM £25. Tel. (0525) 715013.

Master Turbo 128, Microvitec monitor DP35, disc drives and single 3.5" disc drive, 40 3.5" discs 90 5.25" discs, disc boxes, mouse, Wordwise plus, Spellmaster, Intersheet, System Delta Plus, ROM manager, Pagemaker, ROM manager, View, Viewsheet, Watford Utilities, Gemini Mailist. Any offers? £650 complete. Tel. (0727) 47601.

BBC model B, Acorn DFS, ATPL Sidewise ROM board, Watford Elec. 32k Shadow RAM board, Microvitec Cub 653 monitor, Acorn single disc drive, pair of Voltmace Delta B joysticks, Interword, Intersheet and Interbase on ROM, Elite on disc, about 30 floppy discs. Full documentation etc. for the software £330. Tel. (0325) 333333.

Sinclair ZX81 computer, 16k RAM, mains adaptor, handbook, offers please. Wordwise ROM £7, BEEBUG Toolkit ROM £7, Dr Who & Mines of Terror ROM & disc £7, Basicode 2+ & programs £7, BEEBUG Spellcheck disc £5, BEEBUG Masterfile disc £5, Superior Software Speech! £7, Cheetah Speech Synthesis module £7, all with handbooks. Tel. (0262) 677555 (work) (0377) 42037 (home).

Logistix, SoliCAD, Graphic Writer, System Delta Plus, Conqueror game. Offers? Tel. 01-540 8461.

Microlink Multi-speed Modem (V21/22/23), 32 number store, auto-answer/dial, BBC Micro cable, Comms software, manuals. Unrequired gift, cost £179, sell £149. Tel. (0952) 581407.

BEEBUG magazines, complete from first issue to current with indexes, vols 1 to 6 in binders, remainder loose, Offers? Tel. (044282) 4543.

BBC B v.g.c, cables, £250 of quality software,educational/adventure/entertainment, tape recorder plus cables, nearly new Voltmace Delta 3b twin joysticks manuals, books + BEEBUG magazines 1985-1990 all original. Worth over £700. £250 o.n.o. Tel. (0473) 623448.

Aries B32 shadow RAM board £40, Aries B12 ROM board with 2 x 16k SWR £25, Interword (as new) £25, PMS Multifont NTQ £20, extra fonts for NTQ (2 discs) £8, Cumana CS100 disc drive with PSU £40, Viglen console for BBC B with heavy duty PSU for two disc drives £25, all as new and complete with instructions/manuals, Masterfile II: Qucikcalc: Imogen: Fortress: Ravenskull: Speech! all BBC 40 track disc originals with manuals £4 each. Elite - BBC 40T disc, as new with manuals £8, Graphic Adventure Creator, as new with manuals £10. Tel. (0325) 463873 eves.

BBC software, Watford ROMs: ROMspell, ROM Manager, . BEEBfont, TransfeROM @ £10 each. BEEBUG discs: Qucikcalc, Design @ £8 each, UIM game (brand new) £14, Machine Code Tutor (2xcass) £4, case w/leads for 5.25" floppy drive (brand new) £10. Tel. (0923) 245537.

Panasonic KXP1081 printer £85, Tandy daisy wheel printer with tractor feed £225, Acorn Z80 second processor £120, Torch Z80 second processor £40, AMX Stop Press with mouse for Master £35, Master manuals 1&2 £6 each, ROM holders for Master, various £4 each, Taxan Green screen monitor £30, Watford Apollo Modem with software £50, Datachart modem 1223 with software £30. Also many other BBC bits and pieces inc. books. Tel. (0223) 872178.

Prism modem & ROM £25, BBC second processor £20. Tel. 01-444 0915.

Cumana dual 40/80 disc drive £150. Tel. (0223) 313385.

Archimedes A440 without monitor, RISC OS, 4Mb RAM 20Mb hard disc, software £1,550 or reasonable offer. CP80 printer £50. Tel. (0272) 736237.

Torch computer (model CF 240M) with Hi-Res colour monitor and dual 720k 5.25" DS DD built in, separate keyboard. Contains complete BBC circuit board with DFS & 16K sideways RAM in addition to native Z80 based CP/M board. Built in COMMS hardware. With manuals and Torch software including Perfect Writer, Perfect Filer, Perfect Calc, Wordstar £350 plus postage or collect. Tel. (0730) 67486 (Hants) eves or weekends.

M128 with 5.25" dual drive (self powered) and mono monitor, hardly used £450. Tel. (0707) 269245.

Monitor: Microvitec 1431 TTL RGB input, metal case as new £149. Tel. (0943) 464824.

WANTED: Instant Mini Office II on ROM. Tel. (0929) 424175.

Inter-Word ROM in original packaging, mint condition, never used £30, Also 25+ BBC games on cassette inc. Citadel, Castle Quest, Moon Cresta, whole lot £35. Tel. 01-229 5449.

BBC B issue 7, control Data DS 40/80 drive, manual and leads £270 o.n.o. Tel. 061-766 6007.

WANTED: Pens for an Epson H1-80 "drawing machine", or name/address of a supplier. Tel. (0903) 40531 eves.

M128 with View, Philips monitor, Viglen Teac 5.25" dual DD, switchable 40/80 with integral power supply, all hardly used. Also CC Mega 3 ROM in Care Master Quad ROM cartridge, unused. Also, Pace Linnett 1200 modem and Commstar II communications software and PSU, unused. All complete with manuals and cables. Tel. (0635) 47082.

Master 512 ver 2.1 Gem software + mouse, Dabs manual + disc, Acorn D/Rec. joystick, Plinth mounted 2X 5.25" DD/DS 40/80 SW D/drives, Smart cartridge, Morley Teletext adaptor + design 7, 1Mb MOS ROM, BEEBUG CAD disc, BEEBUG mags & binders, 5.25" DS DD D/drive aux p/s, tape and disc games, Elite, Exile, Thor etc. Master/Plinth, manuals, leads, dust cover £650 inc. P&P. Tel. (0444) 811716.

Multiprom EPROM programmer, for 8k, 16k, 32k, 64k, 128k, as new £27, Dumpout 3 ROM £17, Viewspell ROM £17, ADI ROM £12, all with manuals and P&P included. WANTED: Master system ROM, Morley AA board, Master circuit board or Master for spares. Tel. 051-647 5367.

BBC B with DFS, 2 drives, ROM board stuffed with goodies, colour monitor, printer, modem, software inc. games, BEEBUG from issue 1 and more £550 the lot, might split. Tel. 01-428 2841 eves and w/e.

Acorn 6502 second processor with DNFS and Hi-Basic ROMs, manual and original packing £65, Sleuth, Printmaster, Source Manager and Prestel ROMs £5 each, 50 BBC games cassettes £70, Freefall, Elite, Revs etc. on discs £4 each. Tel. (0782) 314053 eves and w/e.

Acorn Electron with Slogger ROM box (+ utility ROMs, eg. Addcom, Elkman T2P3), Plus 1 (+ View, Viewsheet cartridges), Plus 3 (+ over 20 discs with various programs), two good joysticks, various commercial software (Pascal, Repton 3, etc.) all equipment in excellent condition and fully documented £180 o.n.o. Tel. (0734) 751875 after 5.30pm.

BBC B OS 1.2 with over 200 games, tape recorder and leads, lots of magazines and 3 vols of BEEBUG. Bargin at £210. Tel. (0932) 242960 7-9pm.

Archimedes 310 entry system, RISC OS, Acorn 2-slot back plane, recently overhauled, original packaging, manuals and discs £600. Tel. (0206) 841119.

Acorn Master cartridges £7 each, Care Quad cartridges £10 each, Original double Acorn joysticks £13, ACP Toolkit ROM £22, New Master ref. manuals 1&2 £10 each, New BBC Advanced User Guide £10, Sharewatch ROM £30. Tel. 01-989 2666.

Master 128, dual 40/80T drives in Viglen "PC" console, Microvitec monitor, AMX mouse, joystick, EPROM programmer and eraser, Overview, Elite, Citadel £650. Juki 6100 daisywheel printer £200. Tel. 01-977 9629.

Cannon laser printer cartridge slightly used, no longer required, any reasonable offer. Tel. (0299) 896845.

M128 monochrome monitor, twin 40/80 DD, Panasonic printer, Interseries ROMs, all hand books £550 o.n.o. Tel. (0276) 22031.

Redboxes: a number of "Red Boxes", together with manuals, no longer required, any offers? 1 red leader, 2 red one, and 4 red two are available. Will sell as one lot. Tel. 01-223 6051 eves.

Morley 2Mb RAM disc plus Vu-Fax software, instant loading and saving brilliant! cost £400 accept £200 o.n.o. must sell, also over 50 software packages; games, music, utilities, for details Tel. 091-529 4788.

M128 complete word processing system, Mini Office II, Mini Office II Utilities Disc + spell checker £20, The Last Ninja (karate game) £5, (all on disc), Bullseye quiz game on cassette £2, Master 512 shareware vol. 1 unused £20, 1987 back issues of Micro User and Acorn User £10. Tel. (0326) 240734 after 6pm.

BEEBUG's Basic debugger Sleuth version 1.05a as new £7 o.n.o. inc. p&p. Tel. (0962) 61102 after 7pm and w/e.

Seikosha GP250X printer currently in use, offers? Tel. (0932) 783387.

Shinwa CP80 printer £80 plus postageor collect (Exmouth). Tel. (0395) 263638.

Defected to IBM, so have Master 512 surplus to requirements, no monitor or disc drives, but includes software, books, etc. for both M128 and M512 £330 o.n.o. for quick sale. Complete set BEEBUG magazines in binders, FREE! you pay postage. WANTED: User manual for IBM Proprinter Model 4201. Tel. (0978) 759732 eves/wkends.

WANTED: M128 Compact with or without monitor, in good condition. Tel. (0753) 651990.

Epson MX-80 FT printer, hardly used £85 o.n.o. Trackerball with drawing package £15, Grafpad MK2 with drawing software £45 o.n.o. Chiptester digital integrated circuit tester, fits BBCB or Master complete with software and handbook £60 o.n.o. also, not computerware but useful Roneo electric duplicator with spares, make me an offer! Tel. (0635) 297701.

M128, Turbo, 512 co-processor, dual 40/80 disc drive, Overview, View Professional, Master ROM, Reference manuals £670. Tel. (0709) 375135.

BBC B with OS.2 includes ADFS, viewsheet viewstore, diagram II, teletext, Advanced disc doctor ROMs etc. 2x40/80 disc drives plus teletext adaptor, co pro adaptor + DOS + Gem, mouse, trackerball and joystick, over 200 discs and cassettes including games, printmaster, wordwise, view word processing packages, AMX etc. books and magazines, offered complete at £950. Tel. (0259) 216555 eves.

Wordwise + ROM complete with two manuals and tutor tape £25, Intersheet (32K ROM) complete with two manuals and reference card £25. Tel. 01-348 1500 after 7pm and weekends.

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES

		BEEBUG & RISC USER
£16.90	1 year (10 issues) UK, BFPO, Ch.I	£25.00
£24.00	Rest of Europe & Eire	£36.00
£29.00	Middle East	£43.00
£31.00	Americas & Africa	£46.00
£34.00	Elsewhere	£51.00

BACK ISSUE PRICES (per issue)

Volume	Magazine	Tape	5"Disc	3.5"Disc
1	£0.40	£1.00	-	-
2	£0.50	£1.00	-	-
3	£0.70	£1.50	£3.50	-
4	£0.90	£2.00	£4.00	-
5	£1.20	£2.50	£4.50	£4.50
6	£1.30	£3.00	£4.75	£4.75
7	£1.30	£3.50	£4.75	£4.75

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. Note that there is no VAT in magazines.

POST AND PACKING

Please add the cost of p&p when ordering individual items. See table opposite.

Destination	First Item	Second Item
UK, BFPO + Ch.I	60p	30p
Europe + Eire	£1	50p
Elsewhere	£2	£1

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 40303, FAX: (0727) 60263
Manned Mon-Fri 9am-5pm
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: Alan Wrigley
Technical Assistant: Glynn Clements
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

Printed by Newnorth-Burt Ltd (0234) 41111 ISSN - 0263 - 7561

Magazine Disc/Cassette

MARCH 1990 DISC/CASSETTE CONTENTS

KEYWORD HIGHLIGHTER - A utility which allows you to list programs with all keywords and REMs highlighted.

ORDER OUT OF CHAOS - Three programs which allow you to investigate the theory of chaos, by exploring the changes in a population caused by different rates of growth.

DICHOTOMOUS KEYS (Part 2) - Last month's program for designing expert systems, extended to include some powerful editing techniques.

USING THE ROM FILING SYSTEM (Part 2) - The second program in this sequence allows you to load files to sideways RAM using the ROM header. The ROM header program from part one is also included for convenience.

STATISTICS FOR PLEASURE - A program which displays histograms for demonstrating Gaussian (normal) distribution of functions of more than one variable.

PRINTING CHARACTERS 128 to 255 - This program allows you to reproduce the Master's extended character set on a model B and print it on both machines.

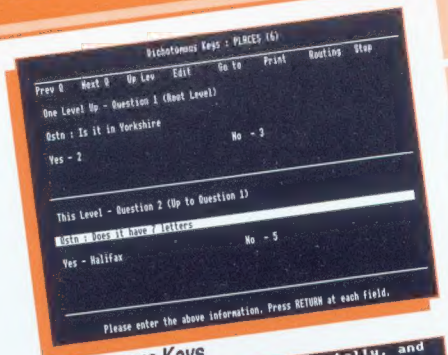
EDIKIT (Part 3) - An addition of five more commands to the EDIKIT ROM.

MASTER DISPLAY ROM (Part 2) - An addition to last month's program (also included on the disc), which allows you to configure the start-up screen colours on your Master.

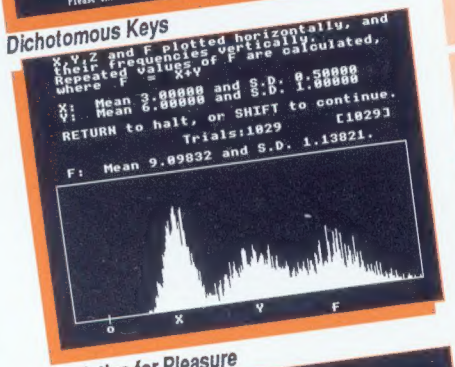
WORKSHOP: WRITING A COMPILER - The complete compiler program discussed in this series.

MAGSCAN DATA - Bibliography for this issue (Vol.8 No.9) and the corrected data for Vol.8 No.8.

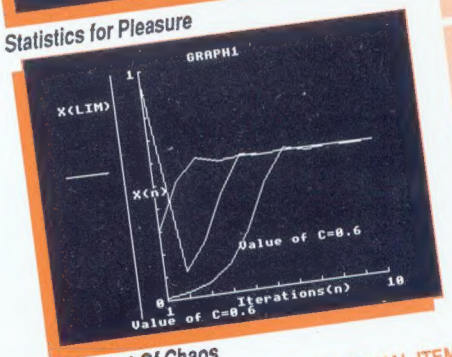
* **FOUR FOURS** - The winning program from our summer competition. *Program on disc only*



Dichotomous Keys



Statistics for Pleasure



Order Out Of Chaos

ALL THIS FOR £3.50 (CASSETTE), £4.75 (5" & 3.5" DISC) + 60P P&P (30P FOR EACH ADDITIONAL ITEM)

Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1, tapes since Vol.1 No.10) available at the same prices.

SUBSCRIPTION RATES	UK ONLY		OVERSEAS	
	Disc (5"or 3.5")	Cassette	Disc (5"or 3.5")	Cassette
6 months (5 issues)	£25.50	£17.00	£30.00	£20.00
12 months (10 issues)	£50.00	£33.00	£56.00	£39.00

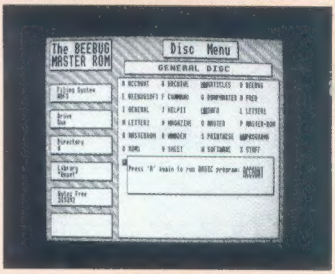
Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:

BEEBUG, 117 Hatfield Road, St.Albans, Herts AL1 4JS.

MASTER ROM

The Beebug Master ROM is a powerful 32K ROM developed to enhance the ADFS, sideways RAM, and real-time features of the Master 128 and Master Compact.



Features include: A comprehensive Disc Front End, Multi-Option Panel, Diary with Calendar and Alarm. Plus a Printer Buffer using sideways RAM, a RAM Disc, and a host of invaluable disc commands.

COMMAND ROM

Disc Menu

Typing *MENU takes you straight to a full feature disc menu and front panel. This gives a readout of current status and displays all the items in the current directory. Just select the file or directory of your choice to be loaded and run. The menu also allows files to be marked for subsequent copying, renaming or deleting

Disc Commands

The Master ROM boasts a wealth of new disc commands to help the user exploit the power of the ADFS with its nested directory structure. Commands include: **`*BACKUP *WIPE *USED *CATALL *FIND *FORMAT *BACKUP *MERGE *DIRCOPY *FCOPY *GOTO`**

Other Features

❑ **Control Panel** - displays all the Master's preset options as well as all available ROMs. The cursor keys are used to move around the panel, and to adjust the state of any items. Current status settings may be saved away to disc for subsequent reloading at any time.

❑ **Printer Buffer** - the Master ROM boasts a 16K-64K full feature printer buffer. This uses sideways RAM in selectable banks, allowing you to print long documents whilst using the computer for another task.

❑ **Diary & Alarm** - the Master ROM allows you to keep a disc-based diary and alarm with reminders for any date in the year.

❑ **RAM Disc** - a simple RAM disc of up to 64K in length. Commands are provided to save programs and memory to the RAM disc.

Members Price

£29.25

Stock Code: 0087C

The Command Driven Communications ROM

Unlike much communications software, this 16K ROM is command driven and has a very powerful extended command set. In addition, for ease of use, all major features are available at the touch of a function key. Because the ROM may be command driven it is exceptionally easy to link commands together in Basic to meet your own individual needs.

❑ **Text Terminal**
Use this terminal to access Telecom Gold and thousands of bulletin boards worldwide. XMODEM file transfer allows you to send files to a friend for the cost of a phone call.

❑ **Telephone Directory**
Set up the name, number and modem configuration of your favourite bulletin boards for easy recall at a later date. No need to remember telephone numbers any more, just type:

`*CALL PRESTEL`

for example, and everything will be done automatically for you - even sign-on strings and passwords are entered.

❑ **Viewdata Terminal**
A full feature Viewdata Terminal giving access to thousands of pages in Prestel and other Viewdata services.

❑ **Viewdata Editor**
A complete teletext editor, with a full range of editing commands, on-screen help, and a pixel editor.

❑ **Modem Compatibility**
The standard version of Command is suitable for the Magic Modem and similar models (Demon, Apollo etc). The Hayes version is suitable for Hayes and other intelligent modems.

Members Price

£29.25

Stock Codes:
Command 0084C
Hayes Command 0073C